ADAPTIVE EVOLUTIONARY MONTE CARLO FOR HEURISTIC OPTIMIZATION: WITH APPLICATIONS TO SENSOR PLACEMENT PROBLEMS

A Dissertation

by

YUAN REN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2008

Major Subject: Industrial Engineering

ADAPTIVE EVOLUTIONARY MONTE CARLO FOR HEURISTIC OPTIMIZATION: WITH APPLICATIONS TO SENSOR PLACEMENT PROBLEMS

A Dissertation

by

YUAN REN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Yu Ding
Committee Members,	Amarnath Banerjee
	Brett A. Peters
	Faming Liang
Head of Department,	Brett A. Peters

December 2008

Major Subject: Industrial Engineering

ABSTRACT

Adaptive Evolutionary Monte Carlo for Heuristic Optimization: With Applications to Sensor Placement Problems. (December 2008)

Yuan Ren, B.E., Tsinghua University, Beijing, China

Chair of Advisory Committee: Dr. Yu Ding

This dissertation presents an algorithm to solve optimization problems with "black-box" objective functions, i.e., functions that can only be evaluated by running a computer program. Such optimization problems often arise in engineering applications, for example, the design of sensor placement. Due to the complexity in engineering systems, the objective functions usually have multiple local optima and depend on a huge number of decision variables. These difficulties make many existing methods less effective.

The proposed algorithm is called adaptive evolutionary Monte Carlo (AEMC), and it combines *sampling-based* and *metamodel-based* search methods. AEMC incorporates strengths from both methods and compensates limitations of each individual method. Specifically, the AEMC algorithm combines a tree-based predictive model with an evolutionary Monte Carlo sampling procedure for the purpose of heuristic optimization. AEMC is able to escape local optima due to the random sampling component, and it improves the quality of solutions quickly by using information learned from the tree-based model. AEMC is also an adaptive Markov chain Monte Carlo (MCMC) algorithm, and is in fact the first adaptive MCMC algorithm that simulates multiple Markov chains in parallel.

The ergodicity property of the AEMC algorithm is studied. It is proven that the distribution of samples obtained by AEMC converges asymptotically to the "target" distribution determined by the objective function. This means that AEMC has a

larger probability of collecting samples from regions containing the global optimum than from other regions, which implies that AEMC will reach the global optimum given enough run time.

The AEMC algorithm falls into the category of heuristic optimization algorithms, and is applicable to the problems that can be solved by other heuristic methods, such as genetic algorithm. Advantages of AEMC are demonstrated by applying it to a sensor placement problem in a manufacturing process, as well as to a suite of standard test functions. It is shown that AEMC is able to enhance optimization effectiveness and efficiency as compared to a few alternative strategies, including genetic algorithm, Markov chain Monte Carlo algorithms, and meta-model based methods. The effectiveness of AEMC for sampling purposes is also shown by applying it to a mixture Gaussian distribution.

To my parents

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Professor Yu Ding, for his guidance during the development of this dissertation and for his support and mentorship throughout my Ph.D. study at Texas A&M University. I cannot appreciate enough for his help in every aspect of my academic study. I would also like to thank Professor Faming Liang, who serves as one of my Ph.D. committee members. I really appreciate his willingness to work out every technical detail with me and his brilliant research ideas which guided me through my Ph.D. research.

I would also like to extend my gratitude to my other Ph.D. committee members, Professor Brett A. Peters and Professor Amarnath Banerjee, for their constructive suggestions and invaluable discussions throughout the development of my Ph.D. research and their careful review during the writing process of this dissertation.

I also owe many thanks to Mr. Qinyan Liu, Dr. Xingchu Liu, and Dr. Dong Liang, for introducing me to Professor Ding and for their continuous help and support that have made my transition from an undergraduate student to a Ph.D. student much easier.

My thanks also goes to Advanced Metrology lab members; my time at Texas A&M University would have never been as pleasant and enjoyable without their friendship.

Last, but not least, I am deeply grateful and appreciative to my parents and Ms. Tian Tian for their love, encouragement, and enormous support. This dissertation would not be possible without them.

NOMENCLATURE

$H(\cdot)$	Objective function to be optimized
$G(\cdot)$	Function of physical constraints
ω	Vector of decision variables
$\mathcal{W}$	Design space
x	Vector of a sample
$\mathcal{X}$	Sample space
d	Dimension of the sample space
$\boldsymbol{Z}^d$	Set of d-dimensional vectors with integer elements
$\boldsymbol{x}$	Population of samples
n	Number of samples in the population $\boldsymbol{x}$
x_i	i-th sample in the population $\boldsymbol{x}$, $i = 1, 2, \ldots, n$
k	Index of iterations
τ	Temperature
t_i	Temperature for the i-th Markov chain
$\boldsymbol{t}$	Vector of temperatures
$p(\cdot)$	Boltzmann distribution
$f(\cdot)$	Target distribution
$Z(t_i)$	Normalizing constant
p_m	Mutation rate
$\boldsymbol{H}^{(k)}$	Set of high performance samples at iteration k
$\mathcal{D}^{(k)}$	Set of samples obtained after iteration k
h	Percentile value of the $H(x)$ values
$a_j^{(k)}$	Lower bound of rules learned by CART in the j-th dimension at iteration k

$b_j^{(k)}$	Upper bound of rules learned by CART in the j-th dimension at iteration k
l_j	Lower bound in the j-th dimension of the sample space
u_j	Upper bound in the j-th dimension of the sample space
$m^{(k)}$	Number of promising regions generated by CART at iteration k
κ	Probability to sample from "promising" regions
r	Probability to sample in each dimension from "promising" regions
$q(\cdot)$	Proposal distribution
$I(\cdot)$	Indicator function
$T(\cdot\|\cdot)$	Transition probability
P_k	Probability of updating "promising" regions
θ	Number of times the data-mining mode is run
ρ	Parameter controlling the decreasing rate of P_k
M	The number of iterations EMC is run
M'	The number of iterations data-mining mode is run
J	Tree size
J_H	Number of terminal nodes associated with high performance samples
$\mathcal{B}_{\mathcal{X}}$	σ-algebra
K	Markov chain kernel
N	The number of manufacturing stations
$\boldsymbol{\alpha}$	State vector
$\boldsymbol{u}$	Input vector
$\boldsymbol{\zeta}$	Process disturbances
$\boldsymbol{\beta}$	Vector of sensor measurements
$\boldsymbol{\eta}$	Sensor noises

$\boldsymbol{R}$	State transition matrix
$\boldsymbol{S}$	Input matrix
$\boldsymbol{V}$	Observation matrix
$\boldsymbol{\Sigma}$	Covariance matrix
S	Sensitivity function
$\boldsymbol{\pi}$	A matrix transformation
$\lambda_{\min}(\cdot)$	The smallest eigenvalue of a matrix
$H_S(\cdot)$	Shekel's Foxhole function
$H_R(\cdot)$	Rastrigin's function
$H_G(\cdot)$	Griewank function

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

A. Motivation and Problem Statement

Optimization problems arise in many applications. To solve them, one needs to find a solution that gives a maximum or minimum objective function value from a set of feasible solutions. One type of optimization problems involves a "black-box" objective function, i.e., a function that can only be evaluated by running a computer program. The objective function could depend on a large number of decision variables. Without much information about the structure of the objective function and the relationship between the function and the decision variables, there is hardly an obvious way to find the optimal solution.

Although difficult to solve, such optimization problems are often encountered in engineering applications. Due to the complexity of the engineering systems, the "black-box"-type objective function is usually nonlinear, nonconvex, nondifferentiable, and discontinuous, and contains multiple local optima. Without a clear structure in the objective function, engineers often randomly evaluate some solutions and select the best one among them. In most real-world applications, however, there could be an overwhelmingly large number of solutions to be evaluated, making an exhaustive search infeasible. Moreover, the objective function could be computationally expensive, which indicates that one might only be able to afford to evaluate a relatively small number of solutions.

A typical example of such engineering optimization problems is sensor placement application, which motivates this dissertation. Recent advancement in sensor tech-

---

The journal model is *IEEE Transactions on Automatic Control.*

nology allows engineers to distribute multiple sensors at multiple locations to better monitor engineering systems. Such an infrastructure of system-wide deployment of sensors is called a distributed sensor system. MIT's *Technology Review* has identified the distributed sensor system as one of the top ten technologies that will change the world [1]. The problem of optimal sensor placement is crucial because it is at the foundation of making a distributed sensor system effective. A poorly designed system is likely to generate irrelevant, redundant, or even conflicting information, making subsequent information processing difficult or even futile.

Simply put, a sensor placement problem requires one to determine the number and locations of multiple sensors so that certain design criteria can be optimized within a given budget. Sensor placement issues are encountered in various applications, such as manufacturing quality control [2], structural health monitoring [3], transportation management [4], and security surveillance [5]. Depending on the applications, the design criteria to be optimized include, among others, sensitivity, detection probability, and coverage. Naturally, a design criterion is a function of the number and locations of sensors but is of a complicated, usually nonlinear, functional form. Evaluating the design criterion necessitates running a computer program, qualifying it as a "black-box" objective function.

Mathematically, the sensor placement problems can be formulated as a constrained optimization problem.

$$\min_{\omega \in \mathcal{W}} H(\omega) \quad \text{subject to } G(\omega) \geq 0, \tag{1.1}$$

where $\mathcal{W} \subseteq \mathbf{R}^d$, d is the number of decision varibles, ω is a vector of decision variables (i.e., coordinates of sensor locations), $H : \mathcal{W} \to \mathbf{R}$ is a deterministic function (e.g., a user-specified design criterion) to be optimized, and $G(\cdot) \geq 0$ represents physical constraints associated with the engineering systems. Taking sensor placement in an

assembly process for an example, $G(\cdot) \geq 0$ means that sensors can only be installed on the surface of subassemblies, and $H(\cdot)$ represents a certain engineering design criterion [2]. In Chapter V, we will visit this sensor placement problem with more details.

In many cases, the physical constraints are complicated and thus difficult to handle in an optimization routine. Engineers often discretize the sample space $\mathcal{W}$ and create a finite (yet possibly huge) number of solution candidates that satisfy the constraints; please see [6] for an example. For the sensor placement problems, this means to identify all the viable sensor locations *a priori*; this can be done relatively easily because individual sensors are located in a low (less than or equal to three) dimensional space. One should use a high enough resolution for discretization so that "good" sensor locations are not lost. After the discretization, the formulation (1.1) is reduced to an unconstrained optimization problem as in (1.2),

$$\min_{x \in \mathcal{X}} H(x), \tag{1.2}$$

where $\mathcal{X}$ is the sample space that contains a finite number of feasible candidate sensor locations. Clearly, $\mathcal{X} \subset \mathbf{Z}^d$ ($\mathbf{Z}^d$ is the set of d-dimensional vectors with integer elements) is a discrete and finite set. Note that $H(\cdot)$ in (1.2) is still calculated according to the same design criterion as in (1.1), but defined on $\mathcal{X}$. Recall that $H(\cdot)$ is of the "black-box" type, with potentially plenty of local optima, due to the complex nature of engineering systems.

Solving this discrete optimization problem might seem mathematically trivial because one only needs to enumerate all potential solutions exhaustively and select the best one. In most real-world applications, however, there could be an overwhelmingly large number of potential solutions to evaluate, especially when a high-resolution discretization was performed.

Throughout this dissertation, we assume that $H(\cdot)$ is a deterministic function. As optimization problems with stochastic $H(\cdot)$ functions, there is a large body of relevant literature on simulation-based optimization [7]–[13]. All the aforementioned challenges for optimizing "black-box" functions are existent in simulation-based optimization problems. Another major challenge is the stochastic nature of the simulations. Consequently, the performance of a particular solution cannot be evaluated exactly, but instead has to be estimated. Thus, it is not straightforward to conclusively determine if one solution is better than another. Theoretically, one can improve the estimation quality by repeatedly evaluating one solution until essentially no variance exists. However, this would greatly increase the computational resources needed. In practice, given a limited amount of computational budget, one must balance the amount of computation dedicated to estimating the performance of each solution and the amount dedicated to evaluating more solutions. Although important, these issues are out of the scope of this dissertation.

B. Background

To assess performance of an algorithm, people mainly consider two measures: quality of the solution found by the algorithm and computation time needed to find the solution. An ideal global optimization technique finds the global optimal solutions using very little computation and works on a variety of problem structures. Obviously, we are far from achieving this goal. In practice, one has to take advantage of the structure of the relevant problem in order to devise an effective algorithm. For example, if the objective function is linear, linear programming methods [14] would be a good method to use. If the objective function is convex and is differentiable, then gradient-based methods [15] could work well. For a nonlinear, nonconvex, and "black-box"

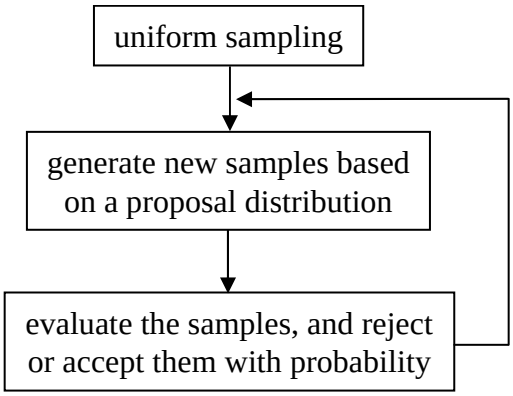


Fig. 1. Flow chart of the sampling-based methods

objective function as considered in this dissertation, more capable algorithms need to be devised.

Generally, two categories of search methodologies have been developed to effectively and efficiently find the best solutions for such a complex, nonlinear optimization problem. The first category is the *sampling-based* methods, and a flow chart of this is shown in Fig. 1. It starts with a set of random samples, and then generates new samples according to some pre-specified mechanism based on current samples. Such a mechanism essentially constructs a proposal distribution that guides the generation of new samples. Then the new samples are accepted/rejected with probability in subsequent iterations. Many well-known optimization methods, such as simulated annealing [16], genetic algorithm [17], and Markov chain Monte Carlo (MCMC) methods [18], fall into this category; the differences among them come from the specific mechanism an algorithm uses to generate and accept new samples. These methods can handle complicated response surface well and have been widely used to solve engineering optimization problems. Their shortcoming is that they generally require a large number of function evaluations before reaching a good solution.

The second category is the metamodel-based methods. Fig. 2 provides a flow

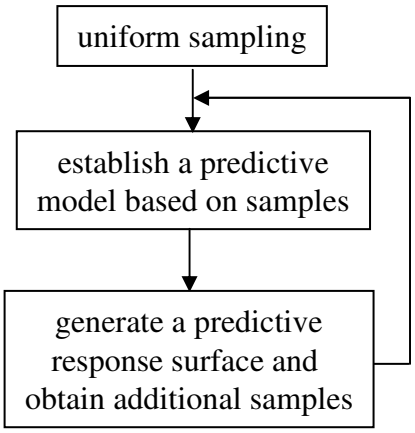


Fig. 2. Flow chart of the metamodel-based methods

chart for this category of methods. It also starts with a set of solution samples $\{x\}$. A metamodel is a predictive model fitted by using the historical solution pairs $\{x,\ H(x)\}$, where $H(x)$ is the value of the objective function evaluated at a sample x. With this predictive model, new solutions are generated based on the model's prediction of where one is more likely to find "good" solutions. Subsequently, the predictive model is updated as more solutions are collected. The model is labeled as "metamodel" because $H(x)$ is the computational output based on a computer model. The *metamodel-based* method originates from the research on computer experiments [19]–[23]. This strategy is also called the "data-mining" guided method, especially when the predictive model used therein is a classification tree model [6],[24],[25], since the tree model is a typical "data-mining" tool. For the metamodel-based or data-mining guided methods, their major shortcoming is their ineffectiveness in handling complicated response surfaces, and as a result, they only look for local optima.

A more detailed literature review on existing algorithms will be provided in Chapter II.

C. Research Objective

The objective of this dissertation is to develop an algorithm that solves optimization problems with deterministic and "black-box" objective functions as formulated in (1.2). The proposed algorithm incorporates strengths from both *sampling-based* and *metamodel-based* search methods and compensates limitations of those individual methods. Therefore, compared to the existing *sampling-based* and *metamodel-based* methods, the proposed algorithm should find better solutions using less computation time. We will use the algorithm to solve the sensor placement problem to show its effectiveness for engineering optimization problems. In order to show its potential for general optimization purposes, we will also test the proposed algorithm on a suite of well known test functions. Furthermore, we will theoretically show the algorithm's ability to reach the global optimum given enough computation time.

The optimization algorithm developed in this dissertation combines the *sampling-based* and *metamodel-based* search methods. Specifically, the proposed algorithm combines evolutionary Monte Carlo [26], [27] and a tree-based predictive model that is updated adaptively. We thus label the proposed algorithm adaptive evolutionary Monte Carlo (AEMC). AEMC incorporates strengths from both evolutionary Monte Carlo (EMC) and metamodeling: the EMC mechanism allows a search to go over the whole sample space and guides solutions toward promising search regions, while the metamodeling mechanism (i.e., the tree-based predictive model) adaptively learns informative rules from past solutions so that the new solutions generated from these rules are expected to have better objective function values than the ones generated from "blind" sampling operators. For this reason, AEMC could potentially find the global optimum of $H(x)$ faster, and our numerical studies to some extent verify this understanding.

AEMC also falls into the category of adaptive MCMC methods. To the best of our knowledge, all adaptive MCMC methods in the literature are based on simulation of one single Markov chain. AEMC is the first adaptive MCMC method that simulates a population of Markov chains in parallel, and thus it can utilize information from multiple chains to improve the convergence rate. Additionally, we also prove that the proposed AEMC algorithm preserves the ergodicity property of Markov chain samples and therefore the global optimum will be reached given enough run time.

D. Outline of the Dissertation

Following this introduction, this dissertation is organized as follows. Chapter II provides a comprehensive literature review on existing methods for optimization problems with "black-box" objective functions and explains how our proposed method relates to previous methods.

Chapter III first describes the general idea of the proposed AEMC algorithm. The algorithm is viewed from two angles: a sampling viewpoint and a metamodeling viewpoint. Then this chapter briefly reviews the EMC algorithm and the data-mining method used in AEMC. The implementation details of AEMC are then described.

Chapter IV studies the ergodicity property of AEMC. It is proved that the AEMC algorithm preserves the ergodicity property of Markov chain samples, which implies global convergence of the algorithm.

In Chapter V, AEMC is employed to solve a sensor placement problem. To show its potential for other optimization problems, AEMC is tested on a suite of well known test functions as well. A sensitivity analysis on several tuning parameters of AEMC is conducted, providing some understandings of the effects of the parameters on the performance of AEMC. AEMC is finally used to sample from a mixture Gaussian

distribution to show its potential for sampling purposes.

This dissertation is concluded in Chapter VI and some future research directions are pointed out.

CHAPTER II

RELATED WORK

In the literature, there are two categories of methodologies that are shown to be effective in solving optimization problems with "black-box" objective functions. They are *sampling-based* search methods and *metamodel-based* methods. These methods are reviewed in Section A and B in this chapter, respectively. There are some other sequential methods that can handle "black-box" objective functions, and they are reviewed in Section C.

A. Sampling-based Methods

Among the *sampling-based* methods, simulated annealing [16] and genetic algorithm [17] have been used to solve optimization problems for quite some time. They use different techniques to generate new random samples. Simulated annealing works by simulating a sequence of distributions determined by a temperature ladder,

$$f_k(x) = \frac{1}{Z_k} \exp\{-H(x)/\tau_k\},\ k = 1, 2, \cdots,$$

where $\tau_1 > \cdots > \tau_k > \cdots$ form a temperature ladder, Z_k is the normalizing constant, τ_1 is reasonably large so that samples all around the sample space can be generated, and $\lim_{k\to\infty} \tau_k = 0$ so that the algorithm focuses on local search around the global optimum. [28] have shown that if the temperature decreases sufficiently slowly (at a logarithmic rate), simulated annealing can reach the global optimum of $H(x)$ with probability 1. However, no one can afford such a slow cooling schedule in practice. People generally use a linearly or geometrically decreasing cooling schedule, but when doing so, the global optimum is no longer guaranteed.

Genetic algorithm uses some evolution operators such as crossover and mutation

to construct new samples. Following natural selection theories, genetic algorithms select parental samples from the current population according to their fitness. The fitter a sample is, the higher is its probability to be selected as a parent. Crossover operators are applied on two parental samples to produce an offspring that inherits characteristics of both the parents. Mutation operators, which randomly change some characteristics of a sample, are occasionally employed to bring variation to the population. From an initial population, genetic algorithm utilizes the selection strategy, the crossover operators, and the mutation operators to make the process evolve toward better populations and eventually find a sample with satisfactory fitness value. Although a general-purpose optimization method, genetic algorithm is known to converge to a good solution rather slowly and lacks rigorous theories to support its convergence to the global optimum.

The MCMC methods have also been used to solve optimization problems [18], [29]–[31]. Even though a typical application of MCMC is to draw samples from complicated probability distributions, the sampling operations can be readily utilized for optimization. Suppose we consider a Boltzmann distribution $p(x) \propto \exp(-H(x)/\tau)$ for some $\tau > 0$. MCMC methods could be used to generate samples from $p(x)$. As a result, the MCMC method has a higher chance of obtaining samples with lower $H(x)$ values. If we keep generating samples according to $p(x)$, we will eventually find samples close enough to the global minimum of $H(x)$. The MCMC methods perform random walks in the whole sample space and thus may potentially escape from local optima given long enough run time.

Recently Liang and Wong [26],[27] proposed a method called evolutionary Monte Carlo (EMC), which incorporates many attractive features of simulated annealing and genetic algorithm into a MCMC framework. It has been shown that EMC is effective for both sampling from high-dimensional distributions and optimization [26], [27].

Because EMC is essentially an MCMC procedure, it guarantees the ergodicity of the Markov chain samples. Nonetheless, it appears that there is still a need and room to further improve the convergence rate of an EMC procedure.

Adaptive MCMC methods have been used to improve the convergence rate of traditional MCMC methods. They attempt to adaptively tune some parameters of proposal distributions while a MCMC procedure is running. For example, [32] and [33] proposed to use regenerative Markov chains and update the proposal parameters at regeneration times; [34] proposed an adaptive Metropolis algorithm which attempts to update the covariance matrix of the proposal distributions using all past samples. Important theoretical advances on the ergodicity of the adaptive MCMC method have been made by [34]–[37].

Table I summarizes the aforementioned literature on *sampling-based* methods and shows how they are different in terms of obtaining samples and accepting/rejecting samples.

B. Metamodel-based Methods

In essence, *metamodel-based* methods are not much different from other sequential sampling procedures guided by a predictive model, e.g., response surface methodology used for physical experiments [38]. The metamodel based method is rooted in research on design and analysis of computer experiments [19]–[23]. In computer experiments, computer models (sets of computer codes) are used to approximate physical experiments. Despite great advances in computing power in the past decades, running such computer codes remains time consuming (e.g, the computer model could be a finite element model). Statistical modeling techniques thus are used to build *inexpensive* surrogates or substitutes of the computer models. Therefore, the statistical model is

Table I. Summary of sampling-based methods

Methods	Generating samples	Accepting samples	Features
Simulated annealing	Using a proposal distribution	According to the Metropolis-Hastings rule	Converges to the global optimum if temperature decreases sufficiently slowly
Genetic algorithm	Using selection, crossover, and mutation operators	Always accept new samples	Population based; no theories about its global convergence
MCMC	Using a proposal distribution	According to the Metropolis-Hastings rule	Obtained samples are asymptotically distributed according to the target distribution
EMC	Construct proposal distribution using selection, crossover, mutation, and exchange operators	According to the Metropolis-Hastings rule	A MCMC algorithm; simulates a population of Markov chains in parallel
Adaptive MCMC	The same as MCMC		Parameters in the proposal distribution are updated as the algorithm is running

actually a "model of the model", or *metamodel* [39], of the computer codes. Various statistical predictive methods have been used to build the metamodels, according to the survey by [19], including neural networks, tree-based methods, Splines, and spatial correlation models.

During the past few years, there have emerged a number of research developments, labeled as data-mining guided engineering designs [6], [24], [25], [40]–[42]. The data-mining guided methods are basically one form of the *metamodel-based* method because they also use a statistical predictive model to guide the selection of design solutions. The predictive models used in the data-mining guided designs include regression [40], classification tree [6], [41], [42], and clustering methods [6].

When looking for an optimal solution, the predictive model is used as follows. After fitting a metamodel (or simply a model, in the case of physical experiments), one could use it to predict where good solutions are more likely to be found and thus select subsequent samples accordingly. This sampling-modeling-prediction procedure is considered a data-mining operation. Kim and Ding [6] and Liu and Igusa [24] demonstrated that the data-mining operation could greatly speed up computation under right circumstances. In [6], a 10 folds time reduction compared to simulated annealing was achieved in their optimal fixture layout design; in [24], a 15 folds time reduction compared to genetic algorithm was achieved in their civil structure optimization. Compared with the slow converging *sampling-based* methods, the *metamodel-based* methods can be especially useful when one has limited amount of data samples; this happens when physical experiments or computer simulations are expensive to conduct. But the metamodel based methods are "greedy" search methods and can be easily entrapped in local optima.

C. Other Sequential Methods

There are several other sequential methods (not metamodel-based) that are applicable for solving optimization problems with "black-box" objective functions.

The gradient-based methods are sequential methods that take advantage of the first and/or second order derivative of the objective function. Some commonly used methods include steepest descent method [15], Newton's method [15], sequential quadratic programming [43], and nonlinear simplex method [44]. For "black-box" objective functions, the gradient needs to be numerically calculated. These methods generally converge to a solution very fast. However, they are all local search methods, i.e., they can not escape from local optima and their performances highly depend on where a search starts.

Recent developments in sequential optimization methods include the NT (number theoretic)-nets based search [45] and the COMPASS (convergent optimization via most-promising-area stochastic search) algorithm [13]. They evaluate a population of solutions uniformly sampled over the sample space at the initial step and over a narrowed space in the subsequent steps, and are therefore less dependent on a single initial starting point. From the current population of solutions, these sequential methods generate new solutions by sampling uniformly from a "promising" region constructed around the current best solution. The NT-nets based search constructs the "promising" region by a rectangular region centered around the current best solution; the COMPASS algorithm defines the "promising" region to be a convex region from which solutions are at least as close to the current best solution as they are to others that have been visited before. Clearly, the "promising" regions in COMPASS are more sophisticated than in the NT-nets based search method; hence we will only include COMPASS in our algorithm performance comparison in Chapter

V. While these methods converge relatively fast, they still suffer from the shortcoming of being easily trapped into local optima.

Note that the COMPASS algorithm is originally designed for simulation-based optimization with stochastic objective functions. An important part of the COMPASS algorithm is to handle the random outputs from a simulation of stochastic nature. But apparently the algorithm is also applicable to optimization problems with deterministic objective functions, which is the case for this dissertation study. So when we compare the performance of AEMC with COMPASS, we simply implement COMPASS for a deterministic objective function $H(\cdot)$.

The proposed AEMC algorithm in this dissertation integrates sampling-based and metamodel-based methods and is distinctively different from those in literature. We will provide more elaborations in Chapter III after the details of AEMC are discussed.

CHAPTER III

ADAPTIVE EVOLUTIONARY MONTE CARLO (AEMC)

A. General Idea of AEMC

The strengths as well as the limitations of the *sampling-based* and the *metamodel-based* search methods spur us to combine the two schemes and develop the AEMC algorithm. The intuition behind how AEMC works is explained as follows.

A critical shortcoming of the *metamodel-based* methods is that their effectiveness highly depends on how representative the sampled solutions are of the optimal regions of the sample space. Without representative data, the resulting metamodel could mislead the search to non-optimal regions. Consequently, the subsequent sampling from those regions will not help the search get out of the trap. In particular, when the sample space is large and good solutions only lie in a small portion of the space, data obtained by a uniform sampling from the sample space will not be representative enough. Under this circumstance, a stand-alone metamodeling mechanism could hardly be effective (as shown in the numerical results in Chapter V), thereby promoting the need to improve the sample quality for the purpose of establishing a better metamodel.

It turns out that *sampling-based* algorithms (we choose EMC in this dissertation), though slow as a stand-alone optimization tool, are able to improve the quality of the sampled solutions. This is because when conducting random searches over a sample space, EMC will gradually converge in distribution to a target distribution that depends on $H(\cdot)$, i.e., the smaller the value of $H(x)$ is, the higher the probability of sampling x is (recall that we want to minimize $H(x)$). In other words, EMC will iteratively and stochastically direct current samples toward the optimal regions such

that the visited solutions are more representative of the optimal regions of the sample space. With the representative samples produced by EMC, a metamodeling operation could generate more accurate predictive models to characterize the promising sub-regions of the sample space.

The above view is made at the vantage point of seeing how the random sampling can help the metamodel-based search methods, namely that the role of EMC in the AEMC method is a quality improver of solution samples obtained from the predictive model. It would be interesting as well to see the benefit of such a hybrid from the random sampling's viewpoint.

The primary demand for improving the *sampling-based* search is to speed up its convergence rate. As argued in Chapter II, making a MCMC method adaptive is an effective way of achieving such an objective. In fact, the proposed AEMC method falls into the category of adaptive MCMC methods; it takes the EMC procedure and makes it adaptive. The metamodel part of AEMC learns the function surface of $H(x)$, and allows us to construct more effective proposal distributions for subsequent sampling operations. As argued in [46], the rate of Markov chain convergence to a target distribution depends crucially on the relationship between the proposal function and the target function $H(x)$. To the best of our knowledge, AEMC is also the first adaptive MCMC method that simulates multiple Markov chains in parallel, while the existing adaptive MCMC methods are all based on simulation of one single Markov chain. So the AEMC can utilize information from multiple chains to improve the convergence rate.

The above discussions explain the benefit of integrating the *metamodel-based* and *sampling-based* method and executing them alternately in a fashion shown in Fig. 3.

For metamodeling (or data-mining) operations, we use classification and regression trees (CART), proposed by [47], to fit predictive models. So more specifically,

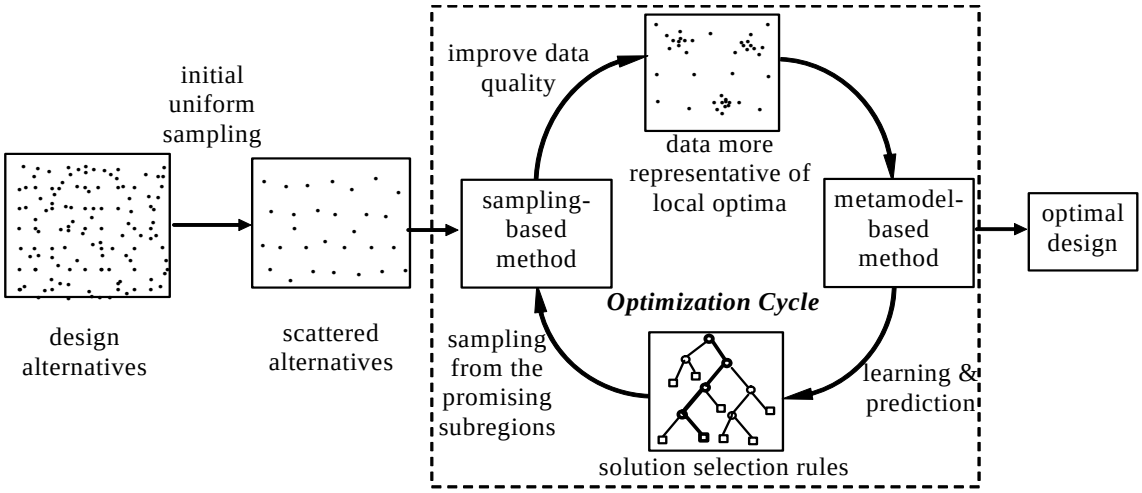


Fig. 3. General framework of combining sampling-based and metamodel-based methods

the AEMC method is a CART-guided EMC. We chose CART primarily because of its computational efficiency. Our goal of solving an optimization problem requires the data-mining operations to be fast and computationally scalable in order to accommodate large-sized data sets. Since the data-mining operations are repeatedly used, a complicated and computationally expensive method will unavoidably slow down the optimization process.

Including the data-mining ingredient is beneficial to engineering optimization also in the sense that in addition to obtaining the optimal solution for the current process configuration, one also garners guidelines about how to construct good solutions, which can facilitate future optimization. Taking sensor placement in an assembly process for an example, if the shape of a part is changed, a previously determined sensor location could become unfeasible. With the guidelines mined from previous design optimization processes, manufacturers could adjust their previous sensor placement without having to run the optimization procedure again from the beginning.

B. Background: MCMC, EMC, and CART

Before presenting the details of the AEMC algorithm, we shall provide some background information in this section so that the AEMC algorithm can be more easily understood.

1. Markov Chain Monte Carlo

As mentioned before, AEMC is an adaptive MCMC algorithm. Some basic knowledge about MCMC is introduced in this section, and the adaptiveness will become clear after AEMC is described. A Markov chain Monte Carlo (MCMC) algorithm is a method of drawing samples from a probability distribution. The idea of MCMC was first introduced by Metropolis et al. [48] for efficient simulation of the energy levels of atoms and was subsequently generalized by Hastings [49] for statistical problems, such as sampling from a probability distribution. The idea of MCMC is as follows. Suppose we have a probability distribution $f(x), x \in \mathcal{X}$ (commonly called a *target distribution*). If $f(x)$ is complex and we can not sample directly from it, an alternative is to construct a Markov chain of samples on $\mathcal{X}$ whose distribution converges asymptotically to the target distribution $f(x)$. Then, if we run the chain for sufficiently long, the simulated samples can be treated as samples from $f(x)$.

A common way to construct the Markov chain samples, $x^{(0)}, x^{(1)}, \ldots, x^{(k)}, \ldots$, is to use the Metropolis-Hastings algorithm [48], [49]. The algorithm updates the Markov chain sample $x^{(k)}$ to $x^{(k+1)}$ as follows. Given the current sample $x^{(k)}$, a sample y is first generated by a probability distribution $q(x^{(k)}, \cdot)$. Since the sample y is a *proposal* or a candidate for the next state of the Markov chain, the distribution q is usually called *proposal distribution*. This candidate sample y is accepted as $x^{(k+1)}$

with probability

$$\min\left\{1, \frac{f(y)q(y, x^{(k)})}{f(x^{(k)})q(x^{(k)}, y)}\right\}.$$

If the candidate is rejected, the chain stay unchanged and $x^{(k+1)} = x^{(k)}$. This rule of accepting a candidate sample is called the *Metropolis-Hastings rule*.

It is proved that the distribution of the samples obtained by the Metropolis-Hastings algorithm will converge to the target distribution asymptotically [50], [51]. However, the convergence rate depends crucially on the relationship between the proposal distribution q and the target distribution f [46]. Intuitively, if q is close to f, sampling from q is then similar to sampling from f directly. If q is far from f, the probability of accepting a candidate sample could be very small and thus the chain could stay at one state for a long time. Therefore it is critical to construct an "effective" proposal distribution. Evolutionary Monte Carlo [26], [27] introduced in the next section borrows the "learning" capability of genetic algorithm to construct proposal distributions close to the target distribution.

2. Evolutionary Monte Carlo

For the convenience of reading this dissertation, we provide a brief summary of EMC in this section and a description of the operators of EMC. Please refer to [26] and [27] for more details. We follow the notations used in [26] and [27]. EMC integrates features of simulated annealing and genetic algorithm into a MCMC framework. Similar to simulated annealing, EMC uses a temperature ladder and simultaneously simulates a population of Markov chains, each of which is associated with a different temperature. The chains with high temperatures can easily escape from local optima, while the chains with low temperatures can search around some local regions and find better solutions faster. The population is updated by crossover and mutation operators,

just like genetic algorithm, and therefore adopts some level of "learning" capability, i.e., samples with better fitness will have a greater probability of being selected and pass their good "genetic materials" to the offsprings.

A *population*, as mentioned above, is actually a set of n solution samples. The state space associated with a population is the product of n sample spaces, namely $\mathcal{X}^n = \mathcal{X} \times \cdots \times \mathcal{X}$. Denote a population $\boldsymbol{x} \in \mathcal{X}^n$ such that $\boldsymbol{x} = \{x_1, \cdots, x_n\}$, where $x_i = \{x_{i1}, \ldots, x_{id}\} \in \mathcal{X}$ is the i-th d-dimensional solution sample. EMC attaches a different temperature, t_i, to a sample x_i, and the temperatures form a ladder with the ordering $t_1 \geq \cdots \geq t_n$. We denote $\boldsymbol{t} = \{t_1, \ldots, t_n\}$. Then the Boltzmann distribution can be defined for a sample x_i as

$$f_i(x_i) = \frac{1}{Z(t_i)} \exp\{-H(x_i)/t_i\}, \tag{3.1}$$

where $Z(t_i)$ is a normalizing constant, and

$$Z(t_i) = \sum_{\{x_i\}} \exp\{-H(x_i)/t_i\}.$$

Assuming that samples in a population are mutually independent, we then have the Boltzmann distribution of the population as

$$f(\boldsymbol{x}) = \prod_{i=1}^{n} f_i(x_i) = \frac{1}{Z(\boldsymbol{t})} \exp\{-\sum_{i=1}^{n} H(x_i)/t_i\}, \tag{3.2}$$

where $Z(\boldsymbol{t}) = \prod_{i=1}^{n} Z(t_i)$.

Given an initial population $\boldsymbol{x}^{(0)} = \{x_1^{(0)}, \ldots, x_n^{(0)}\}$ and the temperature ladder $\boldsymbol{t} = \{t_1, \ldots, t_n\}$, n Markov chains are simulated simultaneously. Denote the iteration index by $k = 1, 2, \ldots$. Now we introduce different operators used in EMC.

a. Crossover

From the current population $\boldsymbol{x}^{(k)} = \{x_1^{(k)}, \cdots, x_n^{(k)}\}$, we first select one parental pair, say $x_i^{(k)}$ and $x_j^{(k)}$ $(i \neq j)$. Without loss of generality, we assume $H(x_i^{(k)}) \geq H(x_j^{(k)})$. The first parental sample is chosen according to a roulette wheel procedure with Boltzmann weights. Then the second parental sample is chosen randomly from the rest of the population. Therefore, the probability of selecting the pair $(x_i^{(k)}, x_j^{(k)})$ is

$$\begin{aligned} &\Pr((x_i^{(k)}, x_j^{(k)})|\boldsymbol{x}^{(k)}) \\ &= \frac{1}{(n-1)G(\boldsymbol{x}^{(k)})}\left[\exp\{-H(x_i^{(k)})/\tau_s\} + \exp\{-H(x_j^{(k)})/\tau_s\}\right], \end{aligned} \tag{3.3}$$

where $G(\boldsymbol{x}^{(k)}) = \sum_{i=1}^{n} \exp\{-H(x_i^{(k)})/\tau_s\}$, and τ_s is the selection temperature.

Two offsprings are generated by some crossover operator, and the offspring with a smaller fitness value is denoted as y_j and the other is y_i. All the crossover operators used in genetic algorithm, e.g., 1-point crossover, 2-point crossover and real crossover, could be used here. Then the new population $\boldsymbol{y} = \{x_1^{(k)}, \ldots, y_i, \ldots, y_j, \ldots, x_n^{(k)}\}$ is accepted with probability min(1,r_c),

$$\begin{aligned} r_c &= \frac{f(\boldsymbol{y})}{f(\boldsymbol{x}^{(k)})}\frac{T(\boldsymbol{x}^{(k)}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x}^{(k)})} \\ &= \exp\{-(H(y_i) - H(x_i^{(k)}))/t_i - (H(y_j) - H(x_j^{(k)}))/t_j\}\frac{T(\boldsymbol{x}^{(k)}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x}^{(k)})}, \end{aligned}$$

where $T(\cdot|\cdot)$ is the transition probability between populations, and $T(\boldsymbol{y}|\boldsymbol{x}) = \Pr((x_i, x_j)|\boldsymbol{x}) \cdot \Pr((y_i, y_j)|(x_i, x_j))$ for any two populations $\boldsymbol{x}$ and $\boldsymbol{y}$. The $T(\cdot|\cdot)$ essentially represents a proposal distribution, and in EMC, proposal distributions are constructed based on different operators mentioned in this section. If the proposal is accepted, the population $\boldsymbol{x}^{(k+1)} = \boldsymbol{y}$, otherwise $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$. Note that all the crossover operators are symmetric, i.e., $\Pr((y_i, y_j)|(x_i, x_j)) = \Pr((x_i, x_j)|(y_i, y_j))$. So $T(\boldsymbol{x}|\boldsymbol{y})/T(\boldsymbol{y}|\boldsymbol{x}) = \Pr((y_i, y_j)|\boldsymbol{y})/\Pr((x_i, x_j)|\boldsymbol{x})$, which can be calculated according

to (3.3).

Following the above selection procedure, samples with better $H(\cdot)$ values have a higher probability to be selected. Offspring generated by these parents will likely be good as well. In other words, the offspring have learned from the good parents. So the crossover operator allows us to construct better proposal distributions, and new samples generated by it are more likely to have better objective function values.

b. Mutation

A sample, say $x_i^{(k)}$, is randomly selected from the current population $\boldsymbol{x}^{(k)}$, then mutated to a new sample y_i by reversing the values of some randomly chosen bits. Then the new population $\boldsymbol{y} = \{x_1^{(k)}, \ldots, y_i, \ldots, x_n^{(k)}\}$ is accepted with probability min(1, r_m),

$$\begin{aligned} r_m &= \frac{f(\boldsymbol{y})}{f(\boldsymbol{x}^{(k)})} \frac{T(\boldsymbol{x}^{(k)}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x}^{(k)})} \\ &= \exp\{-(H(y_i) - H(x_i^{(k)}))/t_i\} \frac{T(\boldsymbol{x}^{(k)}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x}^{(k)})}. \end{aligned}$$

The 1-point, 2-point, and uniform mutation are all symmetric operators, and thus $T(\boldsymbol{y}|\boldsymbol{x}^{(k)}) = T(\boldsymbol{x}^{(k)}|\boldsymbol{y})$. If the proposal is accepted, the population $\boldsymbol{x}^{(k+1)} = \boldsymbol{y}$, otherwise $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$.

c. Exchange

Given the current population $\boldsymbol{x}^{(k)}$ and the temperature ladder $\boldsymbol{t}$, we try to change $(\boldsymbol{x}^{(k)}, \boldsymbol{t})$=$(x_1^{(k)}, t_1, \ldots, x_i^{(k)}, t_i, \ldots, x_j^{(k)}, t_j, \ldots, x_n^{(k)}, t_n)$ to $(\boldsymbol{x}', \boldsymbol{t})$ =$(x_1^{(k)}, t_1, \ldots, x_j^{(k)}, t_i, \ldots, x_i^{(k)}, t_j, \ldots, x_n^{(k)}, t_n)$. The new population $\boldsymbol{x}'$ is accepted with probability min(1,r_e),

where

$$r_e = \frac{f(\boldsymbol{x}')}{f(\boldsymbol{x}^{(k)})}\frac{T(\boldsymbol{x}^{(k)}|\boldsymbol{x}')}{T(\boldsymbol{x}'|\boldsymbol{x}^{(k)})}$$
$$= \exp\{(H(x_i^{(k)}) - H(x_j^{(k)}))(\frac{1}{t_i} - \frac{1}{t_j})\}\frac{T(\boldsymbol{x}^{(k)}|\boldsymbol{x}')}{T(\boldsymbol{x}'|\boldsymbol{x}^{(k)})}.$$

If this proposal is accepted, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}'$, otherwise $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$. Typically, the exchange is performed only on states with neighboring temperature values, i.e., $|i - j| = 1$. Let $p(x_i^{(k)})$ be the probability that $x_i^{(k)}$ is chosen to exchange with another state, and $w(x_j^{(k)}|x_i^{(k)})$ be the probability that $x_j^{(k)}$ is chosen to exchange with $x_i^{(k)}$. So $j = i \pm 1$, and $w(x_{i+1}^{(k)}|x_i^{(k)}) = w(x_{i-1}^{(k)}|x_i^{(k)}) = .5$ and $w(x_2^{(k)}|x_1^{(k)}) = w(x_{n-1}^{(k)}|x_n^{(k)}) = 1$. The transition probability $T(\boldsymbol{x}'|\boldsymbol{x}^{(k)}) = p(x_i^{(k)}) \cdot w(x_j^{(k)}|x_i^{(k)}) + p(x_j^{(k)}) \cdot w(x_i^{(k)}|x_j^{(k)})$, and thus $T(\boldsymbol{x}'|\boldsymbol{x}^{(k)}) = T(\boldsymbol{x}^{(k)}|\boldsymbol{x}')$.

Given the operators introduced above, the k-th iteration of EMC consists of two steps:

1. With probability p_m (mutation rate), apply a mutation operator to each sample independently in the population $\boldsymbol{x}^{(k)}$. With probability $1-p_m$, apply a crossover operator to the population $\boldsymbol{x}^{(k)}$. Accept the new population according to the Metropolis-Hastings rule.

2. Try to exchange n pairs of samples $(x_i^{(k)}, x_j^{(k)})$, with i uniformly chosen from $\{1, \ldots, n\}$ and $j = i \pm 1$ with probability $w(x_j^{(k)}|x_i^{(k)})$.

A flow chart of the EMC procedure is given in Fig. 4. The stopping rule in the figure is usually whether a pre-specified number of iterations has been reached. In the crossover operation, the parental individuals are selected in an iterative way, i.e., the parents are chosen from the population which has been updated by previous crossover operations. As described in [26], the crossover operation repeats for $[n/5]$ times, where $[a]$ takes the integer part of a. According to the same source, each time

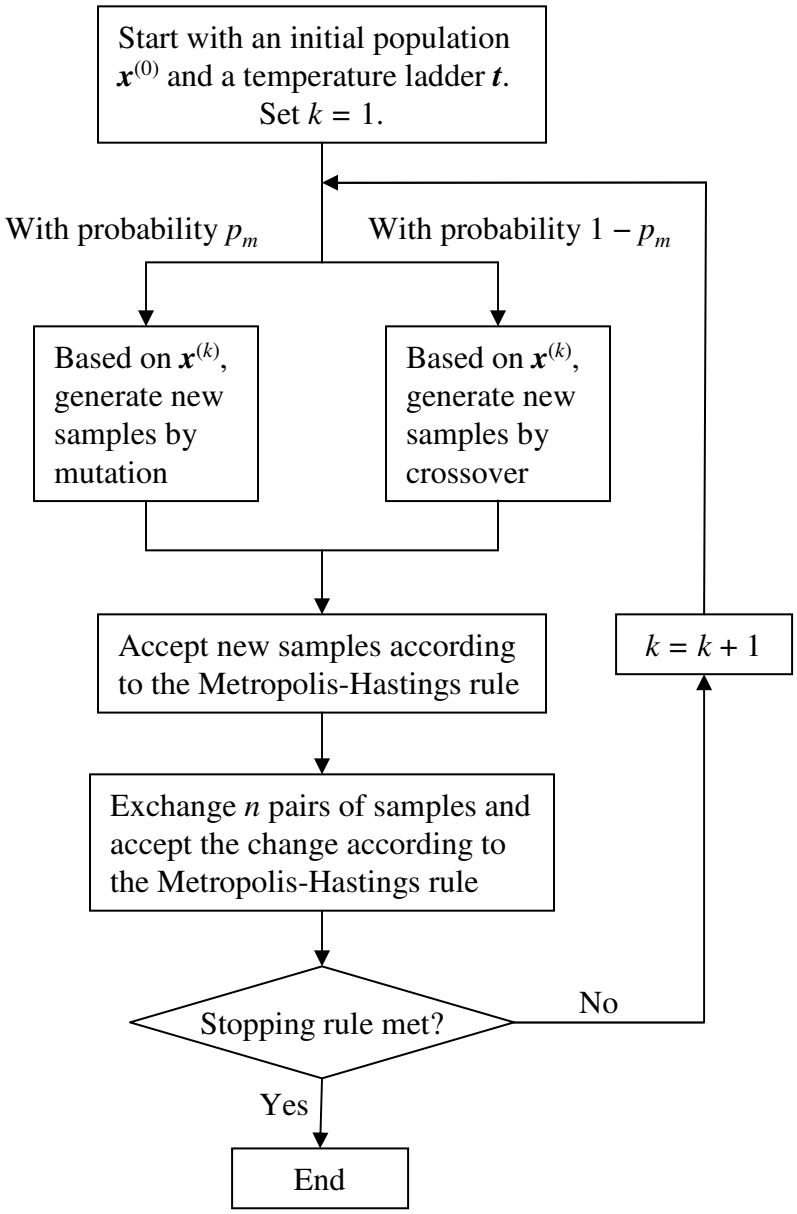


Fig. 4. Flow chart of the EMC algorithm

one applies a crossover operator, about 40% of the individuals in the population are chosen to be parents. Typically, p_m is set to be a small value around .25 for a small or moderate population size (e.g., $n \leq 50$).

EMC is a standard MCMC algorithm and therefore its ergodicity property can be easily verified. Because of the learning capability associated with the crossover operator, EMC could construct proposal distributions more effectively and converge faster than a typical MCMC algorithm.

3. Classification and Regression Trees

Classification and Regression Trees (CART) is a tree-based data mining method proposed by [47]. In this section, we briefly explain how CART fits a statistical model. To facilitate the explanation, we introduce the following notations. There are d predictor variables and one response variable, and we denote the j-th predictor variable by X_j and the response variable by G. Observed values are written in lowercase, and hence the i-th observed values of X and G are written as x_i and g_i, respectively. Denote the number of samples in the data set by L. So the dataset is denoted as (x_i, g_i) for $i = 1, 2, \ldots, L$, with $x_i = (x_{i1}, \ldots, x_{id})$.

CART partitions the sample space into a set of rectangles and fits a simple statistical model in each one. When partitioning the sample space, CART only considers binary partitions. CART first partitions the space into two regions and models the response in each region. It selects the predictor variable and split-point to achieve the best fit. Then one or two of these regions are further split into two more regions. This process is continued until some stopping rule is satisfied. For example, given a square sample space shown in the top panel of Fig. 5, we first split X_1 at point t_1. Then the region $X|X_1 \leq t_1$ is split at $X_2 = t_2$; the region $X|X_1 > t_1$ is split at $X_2 = t_3$. As a result, we get four regions, R_1, R_2, R_3, and R_4. This process is represented by a

binary tree shown in the bottom panel of Fig. 5. Within the m-th region, there are L_m observations associated with it. Let the proportion of class c observations in this region to be

$$\hat{p}_{mc} = \frac{1}{L_m} \sum_{x_i \in R_m} I(g_i = c). \tag{3.4}$$

We then classify the observations in region R_m to class $c(m) = \arg\max_c \hat{p}_{mc}$, the majority class in the region.

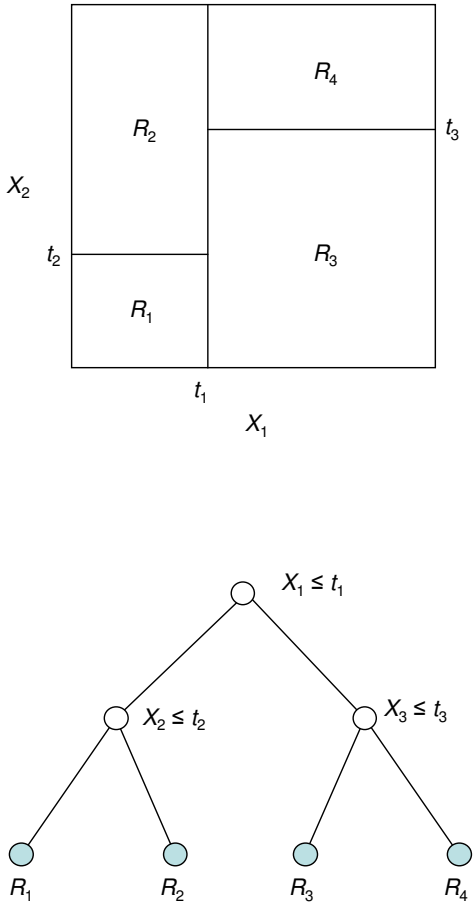


Fig. 5. An example of partitions and CART

A key advantage of CART is its interpretability. From the tree, we can get a set of "if-then" rules that explicitly describe all the regions (e.g., $R_1 \sim R_4$ in

Fig. 5). For instance, the region R_1 is specified by the rules $X_1 \leq t_1$ and $X_2 \leq t_2$. Another advantage of CART is its computational efficiency. The computational complexity of CART is a considerably low $O(dL\log(L))$ [52]. As mentioned before, the computational efficiency and scalability of CART are the main reasons we chose it for our metamodeling purpose.

For more details about fitting a CART model, please refer to [52]. In the sequel, we will present the details of the proposed AEMC algorithm.

C. The AEMC Algorithm

As mentioned before, AEMC contains two modes: EMC mode and data-mining (or metamodeling) mode. We first describe how proposal distributions are learned in the data-mining mode, and then we present the procedures to implement the AEMC algorithm. We shall also discuss selections of parameters involved in AEMC.

1. Data-mining (or Metamodeling) Mode

In AEMC, we first run a number of iterations of EMC and then use CART to learn a proposal distribution (for generating new samples) from the samples produced by EMC. Denote by $\mathcal{D}^{(k)}$ the set of samples we have retained after iteration k. From $\mathcal{D}^{(k)}$, we define high performance samples to be those with relatively small $H(x)$ values. The high performance samples are the representatives of the promising search regions. We denote by $H^{(k)}_{(h)}$ the h percentile of the $H(x)$ values in $\mathcal{D}^{(k)}$. Then, the set of high performance samples at iteration k, $\boldsymbol{H}^{(k)}$, are defined as

$$\boldsymbol{H}^{(k)} = \{x : x \in \mathcal{D}^{(k)} \text{ and } H(x) \leq H^{(k)}_{(h)}\}.$$

As a result, the samples in $\mathcal{D}^{(k)}$ are grouped into two classes, the high performance

samples in $\boldsymbol{H}^{(k)}$ and the others. Treating these samples as a training dataset, we then fit a CART model to a two-class classification problem. Using the prediction from the resulting CART model, we can partition the sample space into rectangular regions, some of which have small $H(x)$ values and are therefore deemed as the promising regions, while other regions as non-promising regions.

The promising regions produced by CART are represented as $a_j^{(k)} \leq x_{ij} \leq b_j^{(k)},\ j = 1, \ldots, d,\ i = 1, \ldots, n$. Since $\mathcal{X}$ is discrete and finite, there is a lower bound l_j and an upper bound u_j in the j-th dimension of the sample space. Clearly we have $l_j \leq a_j^{(k)} \leq b_j^{(k)} \leq u_j$. CART may produce multiple promising regions. We denote by $m^{(k)}$ the number of regions. Then, the collection of the promising regions is specified in the following:

$$\begin{gathered} a_{js}^{(k)} \leq x_{ij} \leq b_{js}^{(k)},\ j = 1, \ldots, d,\ i = 1, \ldots, n, \\ s = 1, \ldots, m^{(k)}. \end{gathered} \tag{3.5}$$

As the algorithm goes on, we continuously update $\mathcal{D}^{(k)}$, and hence $a_{js}^{(k)}$ and $b_{js}^{(k)}$.

After we have identified the promising regions, the proposal density is constructed based on the following thoughts: get a sample from the promising regions with probability κ, and from elsewhere with probability $1 - \kappa$, respectively. We recommend using a relatively large κ value, say $\kappa = .9$. Since there may be multiple promising regions identified by CART, we denote the proposal density associated with each region by $q_{ks}(x)$, $s = 1, \ldots, m^{(k)}$. In this dissertation, we use a Metropolis-within-Gibbs procedure [53] to generate new samples as follows.

For $i = 1, \ldots, n$, denote the population after the k-th iteration by $\boldsymbol{x}^{(k+1,i-1)} = (x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_i^{(k)}, \ldots, x_n^{(k)})$, of which the first $i-1$ samples have been updated, and the Metropolis-within-Gibbs procedure is about to generate the i-th new sample.

Note that $\boldsymbol{x}^{(k+1,0)} = (x_1^{(k)}, \ldots, x_n^{(k)})$.

1. Set S to be randomly chosen from $\{1, \ldots, m^{(k)}\}$. Generate a sample x_i' from the density $q_{kS}(\cdot)$.

$$\begin{aligned} q_{kS}(x_i') = \prod_{j=1}^{d} \Bigg(& r \cdot \frac{I(a_{jS}^{(k)} \le x_{ij}' \le b_{jS}^{(k)})}{b_{jS}^{(k)} - a_{jS}^{(k)}} \\ & + (1-r) \cdot \frac{I(x_{ij}' < a_{jS}^{(k)} \text{ or } x_{ij}' > b_{jS}^{(k)})}{(u_j - l_j) - (b_{jS}^{(k)} - a_{jS}^{(k)})} \Bigg), \end{aligned} \tag{3.6}$$

   where $I(\cdot)$ is the indicator function. Here r is the probability of sampling uniformly within the range specified by the CART rules on each dimension. Since each dimension is independent of each other, we have $\kappa = r^d$.

2. Construct a new population $\boldsymbol{x}^{(k+1,i)}$ by replacing $x_i^{(k)}$ with x_i', and accept the new population with probability $\min(1, r_d)$, where

$$\begin{aligned} r_d &= \frac{f(\boldsymbol{x}^{(k+1,i)})}{f(\boldsymbol{x}^{(k+1,i-1)})} \frac{T(\boldsymbol{x}^{(k+1,i-1)} | \boldsymbol{x}^{(k+1,i)})}{T(\boldsymbol{x}^{(k+1,i)} | \boldsymbol{x}^{(k+1,i-1)})} \\ &= \exp\{-(H(x_i') - H(x_i^{(k)}))/t_i\} \\ &\quad \cdot \frac{T(\boldsymbol{x}^{(k+1,i-1)} | \boldsymbol{x}^{(k+1,i)})}{T(\boldsymbol{x}^{(k+1,i)} | \boldsymbol{x}^{(k+1,i-1)})}, \end{aligned} \tag{3.7}$$

   If the proposal is rejected, $\boldsymbol{x}^{(k+1,i)} = \boldsymbol{x}^{(k+1,i-1)}$.

Now it is clear that the proposal density learned by CART at iteration k, denoted by $q_k(\cdot)$, is defined as

$$q_k(x) = \sum_{s=1}^{m^{(k)}} \frac{1}{m^{(k)}} q_{ks}(x) \quad \text{for all } x \in \mathcal{X}, \tag{3.8}$$

since a "promising" region is randomly chosen from $m^{(k)}$ regions with equal probability as shown in Step 1 of the above Metropolis-within-Gibbs procedure.

The transition probability $T(\cdot|\cdot)$ in equation (3.7) is calculated as follows. Since we only change one sample in each Metropolis-within-Gibbs step, the transition probability can be written as $T(x_i^{(k)} \to x_i'|\boldsymbol{x}_{[-i]}^{(k)})$, where $\boldsymbol{x}_{[-i]}^{(k)} = (x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_{i+1}^{(k)}, \ldots, x_n^{(k)})$. Then we have

$$T(x_i^{(k)} \to x_i'|\boldsymbol{x}_{[-i]}^{(k)}) = q_k(x_i').$$

From equation (3.6) and (3.8), it is not difficult to see that as long as $0 < r < 1$, the proposal is global, i.e., $q_k(x) > 0$ for all $x \in \mathcal{X}$. Since $\mathcal{X}$ is finite, it is natural to assume that $f(x)$ is bounded away from 0 and ∞ on $\mathcal{X}$. Thus, the minorisation condition [51], [54], i.e.,

$$\omega^* = \sup_{x \in \mathcal{X}} \frac{f(x)}{q_k(x)} < \infty,$$

is satisfied. As shown in Section IV.C, satisfaction of this condition would lead to ergodicity of the AEMC algorithm.

2. Algorithm Steps

Now we are ready to present a summary of the AEMC algorithm (an algorithm flow chart is given in Fig. 6), which consists of two modes: the EMC mode and the data-mining (or metamodeling) mode.

1. Set $k = 0$. Start with an initial population $\boldsymbol{x}^{(0)}$ by uniformly sampling n samples over $\mathcal{X}$ and a temperature ladder $\boldsymbol{t} = \{t_1, \ldots, t_n\}$.

2. EMC mode: run EMC until a switching condition is met.

    - Apply mutation, crossover, and exchange operators to the population $\boldsymbol{x}^{(k)}$ and accept the updated population according to the Metropolis-Hastings rule. Set $k = k + 1$.

3. Run the data-mining mode until a switching condition is met.

   - With probability P_k, use the CART method to update the promising regions, i.e., update the values of $a_{js}^{(k+1)}$ and $b_{js}^{(k+1)}$ in equation (3.5).
   - With probability $1 - P_k$, do not apply CART and simply let $a_{js}^{(k+1)} = a_{js}^{(k)}$ and $b_{js}^{(k+1)} = b_{js}^{(k)}$.
   - Generate n new samples following the Metropolis-within-Gibbs procedure mentioned earlier in this section. Set $k = k + 1$.

4. Alternate between the two modes until a stopping rule is met. The algorithm could terminate when the computational budget (the number of iterations) is consumed or when the change in the best $H(x)$ value does not exceed a given threshold for several iterations.

3. Advantages of AEMC

The advantages of the proposed AEMC algorithm become intuitively clear if we compare it to a gradient-based algorithm and the COMPASS algorithm. Fig. 7 gives a graphical illustration of each method. A gradient-based algorithm starts from a single solution point and subsequently only evaluates the gradient at a single solution to determine the next action. There is little doubt that a gradient-based algorithm will not go too far on the response surface of a complicated, nonlinear function. The COMPASS algorithm (and the NT-nets based search [45] as well) improves its ability of escaping local optima since they could start with and subsequently evaluate multiple solutions. However, the COMPASS algorithm uses the current best solution to construct a *single* promising search region, and hence only guarantees to find a local optimal solution. AEMC utilizes the CART method to find *multiple* promising

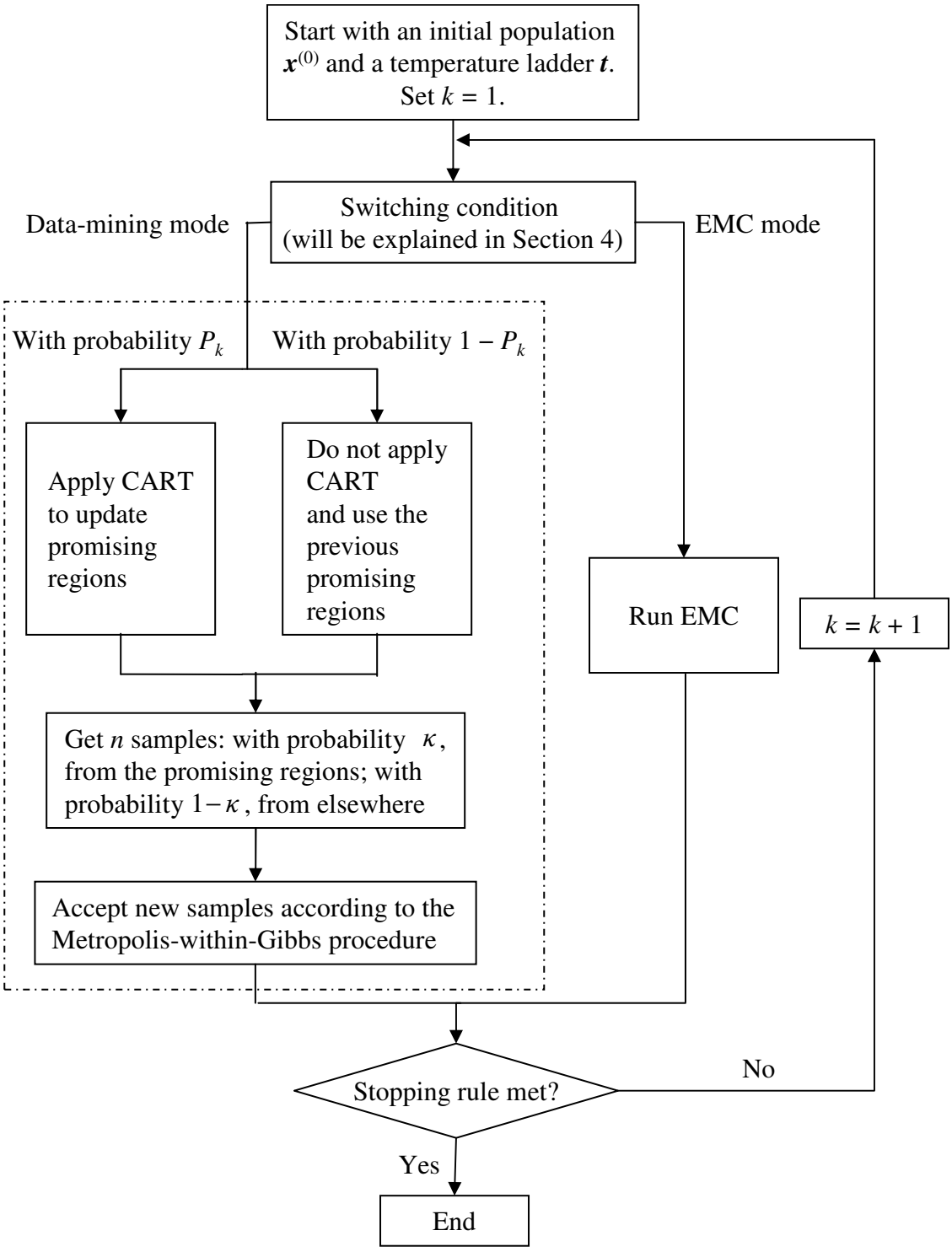


Fig. 6. Flow chart of the AEMC algorithm

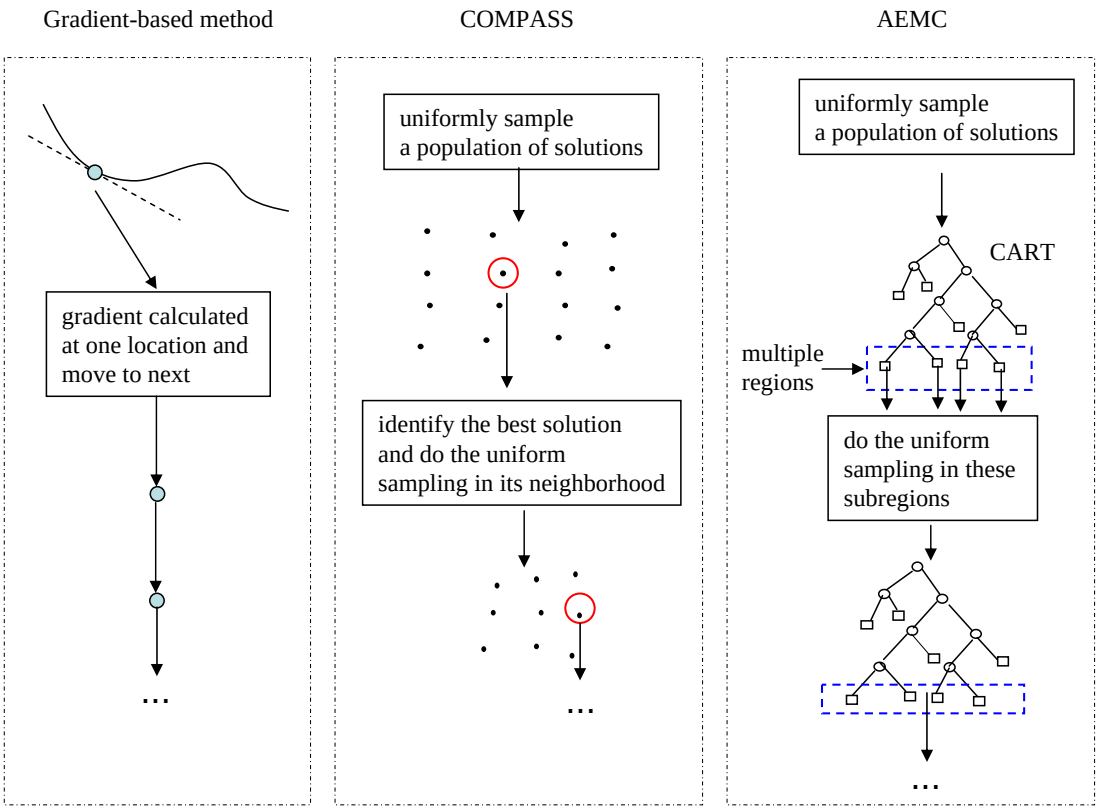


Fig. 7. Comparison of AEMC with other sequential methods

search regions, and therefore has a better chance of finding good solutions. Moreover, since randomness is added from a random sampling component (i.e., EMC), AEMC has the ability of escaping local optima and finding the global optimum.

We also feel that there is some analogy between AEMC and Branch-and-Bound (BB) framework [55]. The BB technique is one of the most popular methods used to solve combinatorial optimization problems. The BB method proceeds by two procedures: branching and bounding. The branching partitions the search space into (ideally) disjoint subregions, and then recursively partitions each of the subregions until no more partition is possible. This generates a tree-structure representation of the search space, called BB tree. The bounding tries to find upper and lower bounds of the minimal values of the objective function over subregions in the BB tree. If the lower bound of a subregion R is greater than the upper bound of any other subregion, then this subregion R and any subregion of R is eliminated from the search. In order for the BB framework to work well, there must be an efficient method to find the upper and lower bounds of objective function values for each subregion. However, given a "black-box" objective function, it is quite challenging to find an efficient way to attain the upper and lower bounds.

In some sense, AEMC also has the *branching* and *bounding* components. In the data-mining operations of AEMC, it uses CART to partition the sample space into different disjoint regions and classifies these regions into two categories: promising regions and unpromising regions. This serves as the *branching* component. Based on the partitions, AEMC then samples from the promising regions with a larger probability than from unpromising regions. To some extent, this serves as the *bounding* component. In AEMC, the *branching* component is data-driven and the *bounding* component is probabilistic. The structure of the objective function is not required *a priori*; rather, it is learned using the samples obtained during the algorithm ex-

ecution. Thus, comparing to the traditional branch-and-bound method, AEMC is more flexible for solving optimization problems with "black-box" objective functions. However, because of the model fitting errors in the data-mining operations, there is always misclassification errors, which may result in imperfect branching and bounding that have to be corrected in subsequent operations. This can be considered as the price paid when flexibility is gained.

4. Parameter Selection

To effectively implement AEMC, several issues need to be considered. Firstly, it is the choice of parameters in EMC: n and p_m. We simply follow the recommendations made in the EMC related research. It is not a surprise that an effective and efficient EMC component will lead to good performance of AEMC. So we set n and p_m to values that favor EMC. Typically, $n = 5 \sim 20$ and $p_m = .25$, and a small n is preferred [26].

Secondly, the choice of P_k. We need to make sure $P_1 > P_2 > \cdots > P_k > \cdots$, and $\lim_{k\to\infty} P_k = 0$, which ensures the diminishing adaptation condition required for the ergodicity of the adaptive MCMC algorithms [37]. As will be discussed in Section IV.C, meeting the diminishing adaptation condition is crucial to the convergence of AEMC. Intuitively, P_k could be considered as the "learning rate". In the beginning of the algorithm, we do not have much information about the function surface, and therefore we apply the data-mining mode to learn new information with a relatively high probability. As the algorithm goes on, we may have sufficient knowledge about the function surface, and it may be a waste to execute the data-mining mode too often. So we make the "learning rate" decrease over time. Specifically, we set $\theta = 0$ at the beginning of the AEMC algorithm. Each time AEMC switches to the data-mining mode, we let $\theta = \theta+1$. So θ denotes the number of times the data-mining mode is run.

Then we set $P_k = 1/\theta^\rho$, i.e., we only update the value of P_k when AEMC switches to the data-mining mode (hence θ depends on both k and the switching condition explained later). The ρ ($\rho > 0$) controls the decreasing speed of P_k. The larger ρ is, the faster P_k decreases to 0. We recommend $\rho = .1$ for general applications.

Thirdly, the construction of training samples $\mathcal{D}^{(k)}$. The question is that should we use all the past samples to construct $\mathcal{D}^{(k)}$ or should we use a subset instead, for example, using only recent samples gathered since the last data-mining operation. If we use all the past samples, data mining will be performed on a large dataset, and doing so will inevitably take a long time and thus slow down the optimization process. Because EMC is able to randomly sample from the whole sample space, AEMC is less likely to fall into local optima even if we just use recent samples. Thus, the latter becomes our choice.

Lastly, we discuss the following tuning parameters of AEMC.

- Switching condition M. In order to adopt the strengths of the two mechanisms and compensate their weaknesses, a proper switching condition is needed to select the appropriate mode of operations for the proposed optimization procedure. One typically runs the EMC mode for M iterations, then switches to the data-mining mode and runs it for M' iterations, and then switches back to the EMC mode, as illustrated in Fig. 8. Because of the inability of a stand-alone data-mining mode to find representative samples (will be shown in Chapter V), it is not beneficial to run the data-mining mode for multiple iterations. So a natural choice is to run the data-mining mode only once (i.e., $M' = 1$) for every M iterations of EMC. If M is too large, the learning effect from the data-mining mode will be overshadowed and AEMC virtually becomes a pure EMC algorithm. If M is too small, EMC may not be able to gather enough representative

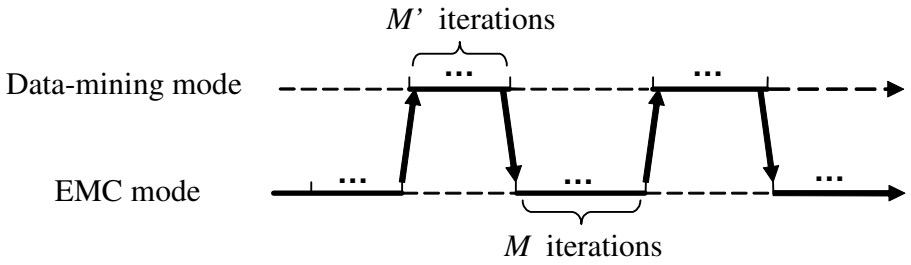


Fig. 8. Switching condition for the AEMC algorithm

data and thus data mining could hardly be effective. We recommend choosing M based on the value of $n \times M$, which is the same size used in the data-mining mode for establishing the predictive model. From our experience based on a sensor placement example, $nM = 150 \sim 450$ works quite well. This means that if n is chosen to be 5, M will take a value between 30 and 90.

- Choice of the percentile value h. For a minimization problem, too large an h value could bring many uninteresting samples into the set of supposedly high performance solutions and then slow down the optimization process. On the other hand, a small value of h would increase the chance for the algorithm to fall into local optimum. Besides, a very small h value may lead to small promising regions, which could make the acceptance rate of new samples too low. But the danger of falling into the local optima is not grave because the data-mining mode is followed by EMC that randomizes the population again and makes it possible to escape from the local optima. In light of this, we recommend an aggressive choice for the h value, i.e. $h = 5\% \sim 15\%$.

- Choice of the tree size in CART. Fitting a tree is to approximate the response surface of $H(x)$. Since we apply CART multiple times in the entire procedure of AEMC, a series of trees, instead of a single tree, is used to approximate $H(x)$.

We believe that the mechanism of how the trees work in AEMC is actually similar to tree boosting, where a series of CART are put together to produce a result. The way people determine right-sized trees for boosting is to consider the degree to which decision variables interact with one another [52]. They argued that a tree with J terminal nodes could model up to $J-1$ levels of interaction effect of different decision variables. Because in most practical applications the low-level interaction effects tend to be dominant, fitting a large tree, i.e., choosing a large J, could overfit the data and cause the resulting model to have a poor prediction power. Hence Friedman [56] recommended that $2 \leq J \leq 10$.

In our problem, however, controlling J alone does not precisely fulfill our objective. Because our goal is to find the global optimum rather than to make good prediction for the whole response surface, we are much more interested in the part of the response surface where the $H(x)$ values are relatively small, corresponding to the class of high performance samples, $\boldsymbol{H}^{(k)}$. It then makes sense to control the number of terminal nodes associated with $\boldsymbol{H}^{(k)}$, denoted by J_H. Controlling J_H enables us to fit a CART model that better approximates a high performance portion of the response surface. Note that the set of terminal nodes representing $\boldsymbol{H}^{(k)}$ is a subset of all terminal nodes in the corresponding tree, meaning that the value of J_H is positively correlated with the value of J. Thus, controlling J_H in the meanwhile also regulates the value of J. The basic rationale behind the selection of J_H is similar to that for J: a large J_H results in a large J and could lead to overfitting; the danger of using too small a J_H is that there will be too few promising regions for the subsequent actions to search and evolve from, which may cause the proposed procedure to miss some good samples. From the above arguments, we note that the $\boldsymbol{H}^{(k)}$ in our

problem plays an analogous role as the whole response surface in the traditional tree boosting. We believe that the guideline for J could be transferred to J_H, i.e., $2 \leq J_H \leq 10$.

In Chapter V, we shall provide a sensitivity analysis of the three tuning parameters, which reveals how the performance of AEMC depends on their choices. From the results, we will see that M and h are the two most important tuning parameters and that AEMC is not sensitive to the value of J_H, provided that it is chosen from the range recommended above.

CHAPTER IV

ERGODICITY OF AEMC

As an adaptive MCMC algorithm, AEMC simulates multiple Markov chains in parallel and could therefore utilize the information from different chains for improving the convergence rate. We will provide some empirical results in support of this claim in Chapter V because a theoretical assessment of convergence rate is too difficult to obtain. But we do investigate the ergodicity property of AEMC, which is relevant to whether the samples obtained by AEMC will converge in distribution to the target distribution.

The ergodicity property implies that AEMC will reach the global optimum of $H(x)$ given enough run time. As we mentioned in Chapter II, if AEMC can generate random samples that are distributed according to the Boltzmann distribution $p(x) \propto \exp(-H(x)/\tau)$, then it has a higher probability to obtain samples with lower $H(x)$ values. If we keep generating samples according to $p(x)$, we will eventually find samples close enough to the global minimum of $H(x)$.

In Section IV.A, some basic knowledge about modeling MCMC algorithms are introduced. In Section IV.B, we will review existing results on the ergodicity of a general adaptive MCMC algorithm. In Section IV.C, we prove a theorem for the ergodicity of the AEMC algorithm.

A. Basics about Modeling MCMC Algorithms

A MCMC algorithm is a method of drawing samples from a target distribution. It constructs a Markov chain of samples, $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k)}, \ldots$, whose distribution converges asymptotically to the target distribution. Following the notations used in Chapter III, we let $f(\cdot)$ be a target probability distribution on a state space $\mathcal{X}^n$ with

$\mathcal{B}_{\mathcal{X}^n} = \mathcal{B}_{\mathcal{X}} \times \cdots \times \mathcal{B}_{\mathcal{X}}$ being the σ-algebra generated by measurable rectangles. Note that we denote the state space here by $\mathcal{X}^n$ because AEMC works on a population of n Markov chains.

Due to Markovian property, we have

$$\begin{aligned} &\Pr(\boldsymbol{X}^{(k+1)} \in \boldsymbol{B} | \boldsymbol{X}^{(k)} = \boldsymbol{x}^{(k)}, \boldsymbol{X}^{(k-1)} = \boldsymbol{x}^{(k-1)}, \ldots, \boldsymbol{X}^{(0)} = \boldsymbol{x}^{(0)}) \\ &\quad = \Pr(\boldsymbol{X}^{(k+1)} \in \boldsymbol{B} | \boldsymbol{X}^{(k)} = \boldsymbol{x}^{(k)}), \quad \forall \boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n}, \end{aligned} \tag{4.1}$$

where $\boldsymbol{X}^{(k)}$ is a $\mathcal{X}^n$-valued random variable representing the state of the Markov chain at iteration k, and $\boldsymbol{B} = B_1 \times \cdots \times B_n$ is a measurable rectangle in $\mathcal{X}^n$, each $B_i \in \mathcal{B}_{\mathcal{X}}$.

The conditional probability shown in Equation (4.1) is essential because it characterizes how a Markov chain evolves over time. In MCMC language, this conditional probability is called *Markov chain kernel*, which represents the probability of the next state of the Markov chain being in $\boldsymbol{B}$ given the current state at $\boldsymbol{x}^{(k)}$. Let $K(\boldsymbol{x}^{(k)}, \cdot)$ denote a Markov chain kernel, i.e.,

$$K(\boldsymbol{x}^{(k)}, \boldsymbol{B}) = \Pr(\boldsymbol{X}^{(k+1)} \in \boldsymbol{B} | \boldsymbol{X}^{(k)} = \boldsymbol{x}^{(k)}), \quad \forall \boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n}. \tag{4.2}$$

There are many ways to construct a Markov chain kernel. A common way is to use the Metropolis-Hastings algorithm [48], [49]. Given the current state of a Markov chain at $\boldsymbol{X}^{(k)} = \boldsymbol{x}^{(k)}$, we begin with a proposal distribution $q(\boldsymbol{x}^{(k)}, \cdot)$ to generate a candidate state $\boldsymbol{y}$. Then the candidate state is accepted as the new state of the chain with probability

$$\mu(\boldsymbol{x}^{(k)}, \boldsymbol{y}) = \min \left\{ 1, \frac{f(\boldsymbol{y}) q(\boldsymbol{y}, \boldsymbol{x}^{(k)})}{f(\boldsymbol{x}^{(k)}) q(\boldsymbol{x}^{(k)}, \boldsymbol{y})} \right\}. \tag{4.3}$$

If the candidate $\boldsymbol{y}$ is accepted, then $\boldsymbol{x}^{(k+1)} = \boldsymbol{y}$. Otherwise $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$. Therefore, the Markov chain kernel $K(\boldsymbol{x}^{(k)}, \boldsymbol{B})$ associated with the Metropolis-Hastings algorithm is the sum of two probabilities as follows. The first probability associates with the situation where the candidate state is accepted. Since the candidate $\boldsymbol{y}$ is gener-

ated by the proposal distribution $q(\boldsymbol{x}^{(k)}, \boldsymbol{y})$ and accepted with probability $\mu(\boldsymbol{x}^{(k)}, \boldsymbol{y})$, the probability of accepting the candidate can be written as $\int_{\boldsymbol{B}} w(\boldsymbol{x}^{(k)}, \boldsymbol{y}) d\boldsymbol{y}$, where $w(\boldsymbol{x}^{(k)}, \boldsymbol{y}) = q(\boldsymbol{x}^{(k)}, \boldsymbol{y})\mu(\boldsymbol{x}^{(k)}, \boldsymbol{y})$. The second probability associates with the situation where the candidate state is rejected and the chain stays unchanged, and it can be written as $\left(1 - \int_{\mathcal{X}^n} w(\boldsymbol{x}^{(k)}, \boldsymbol{z}) d\boldsymbol{z}\right)$. Therefore, the Metropolis-Hastings kernel can be written as

$$K(\boldsymbol{x}^{(k)}, \boldsymbol{B}) = \int_{\boldsymbol{B}} w(\boldsymbol{x}^{(k)}, \boldsymbol{y}) d\boldsymbol{y} + I(\boldsymbol{x}^{(k)} \in \boldsymbol{B}) \left(1 - \int_{\mathcal{X}^n} w(\boldsymbol{x}^{(k)}, \boldsymbol{z}) d\boldsymbol{z}\right), \tag{4.4}$$

where $I(\cdot)$ is the indicator function.

The distribution of samples obtained by the Metropolis-Hastings algorithm will converge asymptotically to the target distribution. Yet the effectiveness of the Metropolis-Hastings algorithm highly depend on how close the proposal distribution $q(\cdot, \cdot)$ is to the target distribution [46]. Without much knowledge about the target distribution, it is difficult to choose an appropriate proposal distribution. Adaptive MCMC algorithms are therefore proposed to tune the proposal distribution while the algorithm is running. However, the convergence property may be destroyed if the proposal distribution is not carefully chosen or updated. Roberts and Rosenthal [37] provided some theoretical results that can be used to guide the choice of a "safe" way to adapt the proposal distribution.

B. Modeling of an Adaptive MCMC Algorithm and Its Ergodicity

In this section, we briefly review the results presented in [37]. We follow the notations and terminologies in [37] to model an adaptive MCMC algorithm. Instead of using one single kernel as in a traditional MCMC, an adaptive MCMC algorithm uses multiple kernels and adaptively updates the kernels during algorithm execution. Let $\{K_\gamma\}_{\gamma \in \mathcal{Y}}$

be a collection of Markov chain kernels on $\mathcal{X}^n$, where γ is the index of different kernels and $\mathcal{Y} = \{1, 2, \dots\}$ is a set of indices. Therefore, $\gamma = 1$ for a traditional MCMC method. For an adaptive MCMC algorithm, γ could be different values at different stages of the algorithm. We let $\Gamma^{(k)}$ be a random variable that determines which Markov chain kernel to use at iteration k of an adaptive MCMC algorithm. So the value of γ is a realization of $\Gamma^{(k)}$.

An adaptive MCMC algorithm updates the state of the chain based on the history of the chain, $\{(X^{(i)}, \Gamma^{(i)})\}_{i=1}^{k}$ (i.e., the history of the chain consists of past samples as well as past kernels). Given the history $\{(X^{(i)}, \Gamma^{(i)})\}_{i=1}^{k}$, an adaptive MCMC algorithm constructs a Markov chain kernel that is used to update the chain, and the conditional distribution of $\Gamma^{(k+1)}$ given $\{(X^{(i)}, \Gamma^{(i)})\}_{i=1}^{k}$ is specified by the particular adaptive algorithm being used. When updating $\boldsymbol{X}^{(k)}$ to $\boldsymbol{X}^{(k+1)}$, the conditional distribution of $\boldsymbol{X}^{(k+1)}$ is

$$\Pr(\boldsymbol{X}^{(k+1)} \in \boldsymbol{B} | \boldsymbol{X}^{(k)} = \boldsymbol{x}, \Gamma^{(k)} = \gamma, \{(X^{(i)}, \Gamma^{(i)})\}_{i=1}^{k-1}) = K_\gamma(\boldsymbol{x}, \boldsymbol{B}),$$
$$\text{for all } \boldsymbol{x} \in \mathcal{X}^n,\ \gamma \in \mathcal{Y},\ \boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n}.$$

Therefore, each kernel K_γ depends not only on the immediate previous state but also on the history of the chain.

Now let us look at how an adaptive MCMC progresses from an initial state until iteration k. Given initial conditions $\boldsymbol{X}^{(0)} = \boldsymbol{x}$ and $\Gamma^{(0)} = \gamma$, we let

$$A^{(k)}((\boldsymbol{x}, \gamma), \boldsymbol{B}) = \Pr[\boldsymbol{X}^{(k)} \in \boldsymbol{B} | \boldsymbol{X}^{(0)} = \boldsymbol{x}, \Gamma^{(0)} = \gamma], \quad \boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n},$$

represent the conditional probability of $\boldsymbol{X}^{(k)}$ for the adaptive MCMC at iteration k. In traditional MCMC methods, Markov chain kernel $K_{\Gamma^{(k)}}$ only depends on immediate previous state $X^{(k-1)}$, and thus we have $A^{(k)} = \prod_{i=0}^{k-1} K_{\Gamma^{(i)}}$. However, in adaptive MCMC methods, because kernels are constructed based on history of the chain, $K_{\Gamma^{(k)}}$

depends on all previous kernels $K_{\Gamma^{(0)}}, \ldots, K_{\Gamma^{(k-1)}}$. So $A^{(k)} \neq \prod_{i=0}^{k-1} K_{\Gamma^{(i)}}$ for adaptive MCMC methods. Actually $A^{(k)}$ is equivalent to integrating over the distributions of $\Gamma^{(1)}, \ldots, \Gamma^{(k-1)}$ and $X^{(1)}, \ldots, X^{(k-1)}$. Then we let

$$\begin{aligned}\Upsilon(\boldsymbol{x}, \gamma, k) &= \|A^{(k)}((\boldsymbol{x}, \gamma), \cdot) - f(\cdot)\| \\ &\equiv \sup_{\boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n}} |A^{(k)}((\boldsymbol{x}, \gamma), \boldsymbol{B}) - f(\boldsymbol{B})|\end{aligned}$$

denote the total variation distance between the $f(\cdot)$ and the distribution of the adaptive MCMC algorithm at iteration k.

After having modeled an adaptive MCMC algorithm, we introduce the definition of ergodicity as follows. It basically says that as the number of iterations of the algorithm goes to infinity, the distribution of samples obtained will converge to the target distribution $f(\cdot)$.

Definition 1. *An adaptive MCMC algorithm is called ergodic if*

$$\lim_{k \to \infty} \Upsilon(\boldsymbol{x}, \gamma, k) = 0, \text{ for all } \boldsymbol{x} \in \mathcal{X}^n \text{ and } \gamma \in \mathcal{Y}.$$

To prove ergodicity of an adaptive MCMC algorithm is not an easy task, and this has been an active research field in the recent years [34]–[37]. Roberts and Rosenthal [37] made a significant step toward providing simpler conditions for the proof. They proved the ergodicity of an adaptive MCMC algorithm under the following three conditions. These results provide a "hunting license" to design ergodic adaptive MCMC algorithms.

(A_1) **[Strongly aperiodic minorisation condition]** There is a $\boldsymbol{C} \in \mathcal{X}^n$, $\varphi > 0$, and for each $\gamma \in \mathcal{Y}$, there exists a probability measure $\nu_\gamma(\cdot)$ on $\mathcal{X}^n$ such that

$$K_\gamma(\boldsymbol{x}, \boldsymbol{B}) \geq \varphi \nu_\gamma(\boldsymbol{B}), \quad \forall \boldsymbol{x} \in \boldsymbol{C}, \ \forall \boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n}, \tag{4.5}$$

(A_2) [**Geometric drift condition**] There is $\mathcal{V} : \mathcal{X}^n \to [1, \infty)$, $0 < \lambda < 1$, $b < \infty$, and $\sup_{\boldsymbol{C}} \mathcal{V} = v < \infty$ such that for each $\gamma \in \mathcal{Y}$

$$K_\gamma \mathcal{V}(\boldsymbol{x}) \leq \lambda \mathcal{V} + bI(\boldsymbol{x} \in \boldsymbol{C}), \quad \forall \boldsymbol{x} \in \mathcal{X}^n \tag{4.6}$$

where $K_\gamma \mathcal{V}(\boldsymbol{x}) = \int_{\mathcal{X}^n} K_\gamma(\boldsymbol{x}, \boldsymbol{y}) \mathcal{V}(\boldsymbol{y}) d\boldsymbol{y}$.

(A_3) [**Diminishing adaptation condition**] The diminishing adaptation condition holds if

$$\lim_{k \to \infty} \sup_{\boldsymbol{x} \in \mathcal{X}^n} \| K_{\Gamma^{(k+1)}}(\boldsymbol{x}, \cdot) - K_{\Gamma^{(k)}}(\boldsymbol{x}, \cdot) \| = 0. \tag{4.7}$$

Given these three conditions, the theorem about ergodicity of an adaptive MCMC stated in Theorem 18 in [37] is as follows.

Theorem 1. *(Theorem 18 in [37]) Consider an adaptive MCMC algorithm with a collection of Markov chain kernels $\{K_\gamma\}_{\gamma \in \mathcal{Y}}$ satisfying the conditions (A_1), (A_2) and (A_3), and $\mathrm{E}[\mathcal{V}(\boldsymbol{x})] < \infty$. Then the adaptive algorithm is ergodic.*

C. Proof of Ergodicity of AEMC

In this section, we will prove that AEMC satisfies the three conditions (A_1), (A_2) and (A_3) and therefore is ergodic. Denote the Markov chain kernel in the data-mining mode at iteration k by $K^{(\mathtt{DM})}_{\Gamma^{(k)}}(\boldsymbol{x}, \cdot)$. To prove the ergodicity of AEMC, we need to prove the conditions (A_1), (A_2) and (A_3) for $K^{(\mathtt{DM})}_{\Gamma^{(k)}}(\boldsymbol{x}, \cdot)$, $k = 0, 1, 2, \ldots$. We denote the proposal density learned by CART by $q_{\Gamma^{(k)}}(\cdot)$, which is given by equation (3.8) (so $q_{\Gamma^{(k)}}(\cdot)$ is equivalent to $q_k(\cdot)$ in (3.8)). For notational simplicity, we drop the subscript ($\mathtt{DM}$) and k afterwards, denoting $K^{(\mathtt{DM})}_{\Gamma^{(k)}}$ by K_Γ, $q_{\Gamma^{(k)}}$ by q_Γ, and $\boldsymbol{x}^{(k)}$ by $\boldsymbol{x}$.

Since a Metropolis-within-Gibbs procedure is used in the data mining mode,

samples $\{x_i\}$ in the population $\boldsymbol{x}$ are updated sequentially. Thus we have

$$K_\Gamma(\boldsymbol{x}, \boldsymbol{B}) = K_\Gamma^{(1)}(x_1, B_1|\boldsymbol{\xi}_1) \times \cdots \times K_\Gamma^{(n)}(x_n, B_n|\boldsymbol{\xi}_n). \tag{4.8}$$

$\boldsymbol{\xi}_i = (y_1, \ldots, y_{i-1}, x_{i+1}, \ldots, x_n)$ denotes the collection of the fixed samples in the transition, where $y_1, \ldots, y_{i-1}$ have been updated and $x_{i+1}, \ldots, x_n$ have not. $K_\Gamma^{(i)}(x_i, B_i|\boldsymbol{\xi}_i)$ is the Metropolis-Hastings kernel for the transition of the i-th sample of the population. Recall equation (4.2), $K_\Gamma^{(i)}(x_i, B_i|\boldsymbol{\xi}_i)$ can be written as

$$K_\Gamma^{(i)}(x_i, B_i|\boldsymbol{\xi}_i) = \int_{B_i} w_\Gamma(x_i, y|\boldsymbol{\xi}_i)dy + I(x_i \in B_i)\left(1 - \int_{\mathcal{X}} w_\Gamma(x_i, z|\boldsymbol{\xi}_i)dz\right), \tag{4.9}$$

where $w_\Gamma(x_i, y_i|\boldsymbol{\xi}_i) = q_\Gamma(y_i)\min\{1, \frac{p(y_i|\boldsymbol{\xi}_i)q_\Gamma(x_i)}{p(x_i|\boldsymbol{\xi}_i)q_\Gamma(y_i)}\}$ and $p(z|\boldsymbol{\xi}_i)$ is a conditional density of $f(\boldsymbol{x})$ with $\boldsymbol{x} = (z, \boldsymbol{\xi}_i)$ ($p(z|\boldsymbol{\xi}_i)$ takes a function form the same as the pdf in equation(3.1)).

First, we prove that the kernel $K_\Gamma(\boldsymbol{x}, \cdot)$ satisfies the strongly aperiodic minorisation condition (A_1).

Lemma 1. *If $0 < r < 1$ and $\mathcal{X}$ is compact, then the Markov chain kernel K_Γ satisfies the strongly aperiodic minorisation condition (A_1).*

Proof. To prove this lemma, we need to find a $\boldsymbol{C}$, $\varphi > 0$, and a probability measure $\nu_\Gamma(\cdot)$ such that the kernel K_Γ satisfies the inequality in (4.5).

Since we have assumed that $\mathcal{X}^n$ is compact, it is natural to assume that $f(\boldsymbol{x})$ is bounded away from 0 and ∞ on the space $\mathcal{X}^n$. As long as $0 < r < 1$, we have that the proposal $q_\Gamma(x)$ is bounded away from 0 due to equation (3.8), and then we have the minorisation condition, i.e.,

$$\omega^* = \sup_{x \in \mathcal{X}} \frac{p(x|\boldsymbol{\xi}_i)}{q_\Gamma(x)} < \infty. \tag{4.10}$$

And then

$$
\begin{aligned}
&K_{\Gamma}^{(i)}(x_i, B_i|\boldsymbol{\xi}_i) \\
&= \int_{B_i} w_{\Gamma}(x_i, y|\boldsymbol{\xi}_i)dy + I(x_i \in B_i)\left(1 - \int_{\mathcal{X}} w_{\Gamma}(x_i, z|\boldsymbol{\xi}_i)dz\right) \\
&\geq \int_{B_i} q_{\Gamma}(y)\min\left\{1, \frac{p(y|\boldsymbol{\xi}_i)q_{\Gamma}(x_i)}{p(x_i|\boldsymbol{\xi}_i)q_{\Gamma}(y)}\right\}dy \\
&= \int_{B_i} \min\left\{q_{\Gamma}(y), \frac{p(y|\boldsymbol{\xi}_i)q_{\Gamma}(x_i)}{p(x_i|\boldsymbol{\xi}_i)}\right\}dy \\
&\geq \int_{B_i} \min\left\{q_{\Gamma}(y), \frac{p(y|\boldsymbol{\xi}_i)}{\omega^*}\right\}dy \qquad \text{(by the equation (4.10))} \\
&= \int_{B_i} \frac{p(y|\boldsymbol{\xi}_i)}{\omega^*}dy \qquad \text{(by the definition of } \omega^*) \\
&\geq \int_{B_i} \frac{p_i^*(y)}{\omega^*}dy \qquad \text{(by defining } p_i^*(y) = \inf_{\boldsymbol{\xi}_i' \in \mathcal{X}^{n-1}} p(y|\boldsymbol{\xi}_i')) \\
&= \frac{p_i^*(B_i)}{\omega^*}.
\end{aligned}
$$

Therefore, we have the following results,

$$
\begin{aligned}
&K_{\Gamma}(\boldsymbol{x}, \boldsymbol{B}) \\
&= K_{\Gamma}^{(1)}(x_1, B_1|\boldsymbol{\xi}_1) \times \cdots \times K_{\Gamma}^{(n)}(x_n, B_n|\boldsymbol{\xi}_n) \\
&\geq K_{\Gamma}^{(1)}(x_1, B_1|\boldsymbol{\xi}_1) \times \cdots \times K_{\Gamma}^{(n-1)}(x_{n-1}, B_{n-1}|\boldsymbol{\xi}_{n-1}) \times \frac{p_n^*(B_n)}{\omega^*} \\
&\cdots\cdots \\
&\geq \prod_{i=1}^{n} p_i^*(B_i)/(\omega^*)^n.
\end{aligned}
$$

Define $\nu_{\Gamma}(\boldsymbol{B}) = \prod_{i=1}^{n} p_i^*(B_i)$ and $\varphi = 1/(\omega^*)^n$, and then we have

$$
K_{\Gamma}(\boldsymbol{x}, \boldsymbol{B}) \geq \varphi\nu_{\Gamma}(\boldsymbol{B}),\ \forall \boldsymbol{x} \in \mathcal{X}^n,\ \forall \boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n}. \tag{4.11}
$$

Equation (4.11) implies that the condition (A_1) is satisfied. □

We then prove that the kernel $K_{\Gamma}(\boldsymbol{x}, \cdot)$ satisfies the geometric drift condition (A_2).

Lemma 2. *If $0 < r < 1$ and $\mathcal{X}$ is compact, then the Markov chain kernel K_Γ satisfies the geometric drift condition (A_2).*

Proof. To prove this lemma, we need to find a function $\mathcal{V}$, $0 < \lambda < 1$, $b < \infty$ such that the kernel K_Γ satisfies the inequality in (4.6).

From equation (4.11), we know that the $\boldsymbol{C}$ needed in condition (A_2) is equal to $\mathcal{X}^n$. Thus, by choosing $\mathcal{V}(\boldsymbol{x}) = 1$, $0 < \lambda < 1$, and $b = 1 - \lambda$, we have

$$\lambda \mathcal{V}(\boldsymbol{x}) + bI(\boldsymbol{x} \in \boldsymbol{C}) = \lambda + (1 - \lambda) = 1$$

and

$$K_\Gamma \mathcal{V}(\boldsymbol{x}) = \int_{\mathcal{X}^n} K_\Gamma(\boldsymbol{x}, \boldsymbol{y}) \mathcal{V}(\boldsymbol{y}) d\boldsymbol{y} = K_\Gamma(\boldsymbol{x}, \mathcal{X}^n) = 1.$$

Therefore, the following condition holds.

$$K_\Gamma \mathcal{V}(\boldsymbol{x}) \leq \lambda \mathcal{V}(\boldsymbol{x}) + bI(\boldsymbol{x} \in \boldsymbol{C}), \ \forall \boldsymbol{x} \in \mathcal{X}^n, \tag{4.12}$$

Then equation (4.12) implies that the condition (A_2) is satisfied. □

Finally, we show the diminishing adaptation condition for the kernel K_Γ.

Lemma 3. *If $\lim_{k \to \infty} P_k = 0$, then the kernel K_Γ satisfies the diminishing adaptation condition (A_3).*

Proof. Suppose at iteration $k + 1$ the newly learned Markov chain kernel by CART is K_{Γ^*}. From the description of the algorithm steps in Section III.2, it is known that with probability P_k, $K_{\Gamma^{(k+1)}} = K_{\Gamma^*}$; otherwise $K_{\Gamma^{(k+1)}} = K_{\Gamma^{(k)}}$. Thus, we have

$$K_{\Gamma^{(k+1)}} = P_{k+1} K_{\Gamma^*} + (1 - P_{k+1}) K_{\Gamma^{(k)}}.$$

Then we have $\|K_{\Gamma^{(k+1)}} - K_{\Gamma^{(k)}}\| = P_{k+1} \|K_{\Gamma^*} - K_{\Gamma^{(k)}}\|$. Since $\lim_{k \to \infty} P_k = 0$, to prove that the kernel K_Γ satisfies equation (4.7), it suffices to prove that $\|K_{\Gamma^*} - K_{\Gamma^{(k)}}\|$ is

bounded. From the definition of a Markov chain kernel in (4.2), we have both $K_{\Gamma^{(k)}}$ and K_{Γ^*} are less than or equal to 1. Therefore $\|K_{\Gamma^*} - K_{\Gamma^{(k)}}\| \leq 1$, and

$$\|K_{\Gamma^{(k+1)}} - K_{\Gamma^{(k)}}\| = P_{k+1}\|K_{\Gamma^*} - K_{\Gamma^{(k)}}\| \leq P_{k+1} \to 0$$

This proves the diminishing adaptation condition. □

Now we are ready to show AEMC is ergodic as long as the proposal $q_\Gamma(\cdot)$ is global and the data-mining mode is run with probability $P_k \to 0$ as $k \to \infty$. This property of the AEMC algorithm is stated in Theorem 2.

Theorem 2. *If* $0 < r < 1$, $\lim_{k\to\infty} P_k = 0$, *and* $\mathcal{X}$ *is compact, then AEMC is ergodic, i.e., the samples* $\boldsymbol{x}^{(k)}$ *converge in distribution to* $f(\boldsymbol{x})$.

Proof. From the proof of Lemma 2, we obviously have $\mathrm{E}[\mathcal{V}(\boldsymbol{x})] = 1 < \infty$. Given the Lemmas 1-3 and Theorem 1, it is apparent that Theorem 2 holds. □

A final note is that we assume the sample space $\mathcal{X}$ to be discrete and bounded in this dissertation. Yet the AEMC algorithm could easily be extended to an optimization problem with a continuous and bounded sample space. One just needs to use the mutation and crossover operators that are proposed in [27]. Theorem 2 still holds. For unbounded sample spaces, we could choose a bounded subspace which is expected to contain the global optimum. Such a choice could be made based on prior experience or expert knowledge.

CHAPTER V

NUMERICAL RESULTS

To illustrate the effectiveness of the AEMC algorithm, we use it to solve three problems: the first two are for optimization purposes and the third is for sampling purposes. The first example is a sensor placement problem in an assembly process. In this example, we shall show sensor placement in two assembly processes: a three-station process and a more complex six-station process. The second example consists of a suite of three well-known test functions for global optimization: Shekel's foxhole function [57], Rastrigin function [58], and Griewank function [59]. In the third example we use AEMC to sample from a mixture Gaussian distribution to see how AMEC, as an adaptive MCMC method, can help with the sampling process.

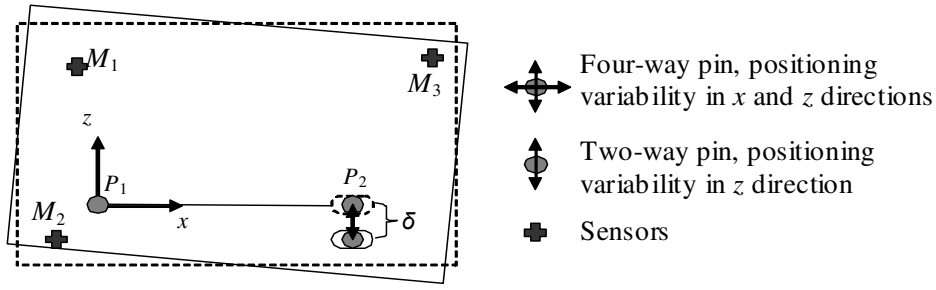
For the two optimization examples, we compare AEMC with EMC, the stand-alone CART guided method, the standard genetic algorithm, and the COMPASS algorithm. About the parameters in AEMC, we chose $n = 5$, $p_m = .25$, $M = 60$, $J_H = 6$ and $h = 10\%$. For the standard genetic algorithm, we let the population size be 100, crossover rate be .9 and mutation rate be .01. For COMPASS, we have set the population size to be 5 (following the recommendation in [13]). All optimization algorithms were implemented in the MATLAB environment, and all reported algorithm performances were the average result of 10 trials. The performance indices for comparison include the best function value found by an algorithm and the number of times that $H(\cdot)$ has been evaluated (also called "the number of function evaluations" hereinafter). The use of the number of function evaluations as a performance measure makes good sense for many engineering design problems where the objective function $H(\cdot)$ is complex and time consuming to evaluate. Thus the time of function evaluations essentially dominates the entire computational cost. In Section V.C, we

will also present a sensitivity analysis on the three tuning parameters, the switching condition M, the percentile value h, and the tree size J_H.

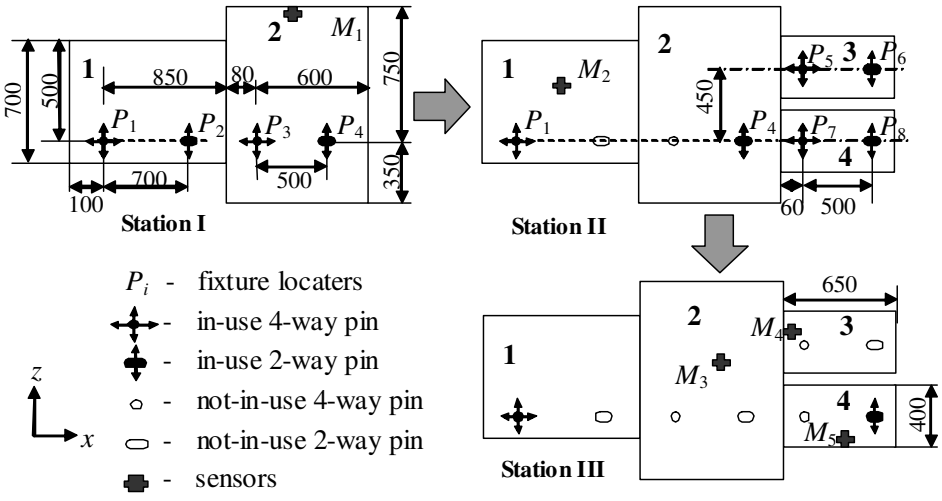
A. Sensor Placement Example

1. Introduction to Sensor Placement in Assembly Processes

We consider a multi-station assembly process, where multiple parts are assembled on multiple stations. Consider as an example the three-station two-dimensional (2-D) assembly process in Fig. 9(b). The three-station assembly process proceeds as follows: (i) at the first station, part 1 and part 2 are assembled; (ii) at the second station, the subassembly consisting of parts 1 and part 2 receives part 3 and part 4; and (iii) at the third station, no assembly operation is performed, but the key dimensional features of the final assembly are measured. The quality assurance objective here is about the dimensional integrity of the assembly product. Optical coordinate sensors are distributed throughout the assembly process to monitor the dimensional quality of the final assembly and/or of the intermediate subassemblies. M_1-M_5 are five coordinate sensors that are currently in place on the three stations; this is simply one instance of, out of hundreds of thousands of other possible, sensor placements. The major variation sources in such a process are associated with the fixture locators on different stations – as shown in Fig. 9(a) – each part or subassembly is held by a set of fixtures, which consists of a four-way pin P_1 that constrains the part motion in both the x and the z directions, and a two-way pin P_2 that constrains the part motion in the z direction. An optimal sensor distribution should be able to identify these fixturing variation sources uniquely and accurately. In order to quantify how well the sensors monitor the variation sources, we first need to understand the relationship between the fixturing variation and the sensor measurements in a multi-station assembly process.



(a) Variation sources on fixture locators



(b) A three-station assembly process

Fig. 9. Illustrative example: a multi-station assembly process

This relation is in fact represented using a station-indexed state space model [60]–[65].

For a multi-station assembly process, the state space model can be expressed as

$$\begin{aligned} \boldsymbol{\alpha}_i &= \boldsymbol{R}_{i-1}\boldsymbol{\alpha}_{i-1} + \boldsymbol{S}_i\boldsymbol{u}_i + \boldsymbol{\zeta}_i \\ \boldsymbol{\beta}_i &= \boldsymbol{V}_i\boldsymbol{\alpha}_i + \boldsymbol{\eta}_i \end{aligned} \tag{5.1}$$

where $i \in \{1, 2, \ldots, N\}$, and N is the number of stations. The state vector $\boldsymbol{\alpha}_i$ and input vector $\boldsymbol{u}_i$ are the product accumulated deviations and the fixture deviation on station i. The process disturbances and sensor noises are denoted by $\boldsymbol{\zeta}_i$ and $\boldsymbol{\eta}_i$, respectively. Product measurements at station i are included in $\boldsymbol{\beta}_i$. In Fig. 9, for instance, $\boldsymbol{\beta}_3$ comprises the deviations measured by sensors M_3 - M_5. The $\boldsymbol{R}_i$ is the state transition matrix which links the part deviation states on adjacent stations, $\boldsymbol{S}_i$ is the input matrix which represents the effect from the fixture deviations, and $\boldsymbol{V}_i$ is the observation matrix corresponding to the number and locations of sensors.

By eliminating the intermediate state variables and aggregating the information associated with individual stations, equation (5.1) could be further formulated into an input-output relation as

$$\boldsymbol{\beta} = \boldsymbol{\Psi} \cdot \boldsymbol{u} + \boldsymbol{\Psi}_0 \cdot \boldsymbol{\alpha}_0 + \boldsymbol{\varepsilon} \tag{5.2}$$

where $\boldsymbol{\beta}^T \equiv [\boldsymbol{\beta}_1^T, \boldsymbol{\beta}_2^T, \ldots, \boldsymbol{\beta}_N^T]$, $\boldsymbol{u}^T \equiv [\boldsymbol{u}_1^T, \boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_N^T]$, $\boldsymbol{\varepsilon}^T \equiv [\boldsymbol{\varepsilon}_1^T, \boldsymbol{\varepsilon}_2^T, \ldots, \boldsymbol{\varepsilon}_N^T]$, $\boldsymbol{\varepsilon}_i \equiv \sum_{j=1}^{i} \boldsymbol{V}_i\boldsymbol{\Phi}_{i,j}\boldsymbol{\zeta}_j + \boldsymbol{\eta}_i$, $\boldsymbol{\Phi}_{i,j} \equiv \boldsymbol{R}_{i-1}\boldsymbol{R}_{i-2}\cdots\boldsymbol{R}_j$, and

$$\boldsymbol{\Psi} \equiv \begin{pmatrix} \boldsymbol{V}_1\boldsymbol{S}_1 & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{V}_2\boldsymbol{\Phi}_{2,1}\boldsymbol{S}_1 & \boldsymbol{V}_2\boldsymbol{S}_2 & \ldots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{V}_N\boldsymbol{\Phi}_{N,1}\boldsymbol{S}_1 & \boldsymbol{V}_N\boldsymbol{\Phi}_{N,2}\boldsymbol{S}_2 & \ldots & \boldsymbol{V}_N\boldsymbol{S}_N \end{pmatrix}, \quad \boldsymbol{\Psi}_0 \equiv \begin{pmatrix} \boldsymbol{V}_1\boldsymbol{\Phi}_{1,0} \\ \boldsymbol{V}_2\boldsymbol{\Phi}_{2,0} \\ \vdots \\ \boldsymbol{V}_N\boldsymbol{\Phi}_{N,0} \end{pmatrix}.$$

Our goal is to detect the *variance change* in the variation sources $\boldsymbol{u}$. For that

purpose, we transform model (5.2) into a variation model. We assume that the product deviation $\boldsymbol{\alpha}_0$, the fixture deviation $\boldsymbol{u}$, and the noise term $\boldsymbol{\varepsilon}$ are independent, and we have

$$\boldsymbol{\Sigma}_{\boldsymbol{\beta}} = \boldsymbol{\Psi} \cdot \boldsymbol{\Sigma}_{\boldsymbol{u}} \cdot \boldsymbol{\Psi}^T + \boldsymbol{\Psi}_0 \cdot \boldsymbol{\Sigma}_{\boldsymbol{\alpha}_0} \cdot \boldsymbol{\Psi}_0^T + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}. \tag{5.3}$$

Note that $\boldsymbol{\Sigma}_{\boldsymbol{u}}$ is a diagonal matrix since the fixture deviations are physically uncorrelated. As such, $\boldsymbol{\sigma} = \text{diag}(\boldsymbol{\Sigma}_{\boldsymbol{u}})$ is the vector of the variances of variation sources. Consider that $\boldsymbol{\Sigma}_{\boldsymbol{\alpha}_0}$ is known from measurements at the end of the precedent fabrication process and that $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$ can be estimated using data from a normal process condition when no outstanding fixture deviation occurs, and then a new covariance matrix is introduced as $\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}} = \boldsymbol{\Sigma}_{\boldsymbol{\beta}} - \boldsymbol{\Psi}_0 \cdot \boldsymbol{\Sigma}_{\boldsymbol{\alpha}_0} \cdot \boldsymbol{\Psi}_0^T - \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$ to summarize the known quantities. Then (5.3) could be rewritten as

$$\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}} = \boldsymbol{\Psi} \cdot \boldsymbol{\Sigma}_{\boldsymbol{u}} \cdot \boldsymbol{\Psi}^T \tag{5.4}$$

Using the $\boldsymbol{\pi}(\cdot)$-transform introduced in [66], equation (5.4) could be expressed as

$$\text{vec}(\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}}) = \boldsymbol{\pi}(\boldsymbol{\Psi}) \cdot \text{diag}(\boldsymbol{\Sigma}_{\boldsymbol{u}}) = \boldsymbol{\pi}(\boldsymbol{\Psi}) \cdot \boldsymbol{\sigma}, \tag{5.5}$$

where $\text{vec}(\cdot)$ is the vector operator [67], and $\boldsymbol{\pi}(\cdot)$ is a matrix transformation defined as

$$\boldsymbol{\pi} : \boldsymbol{\Psi}^{q \times w} \rightarrow \boldsymbol{\pi}(\boldsymbol{\Psi})^{q^2 \times w}$$

$$\boldsymbol{\pi}(\boldsymbol{\Psi}) = \left[(\boldsymbol{\psi}^1 \otimes \boldsymbol{\psi}^1)^T \cdots (\boldsymbol{\psi}^1 \otimes \boldsymbol{\psi}^q)^T \cdots (\boldsymbol{\psi}^q \otimes \boldsymbol{\psi}^1)^T \cdots (\boldsymbol{\psi}^q \otimes \boldsymbol{\psi}^q)^T \right]^T$$

and q, w are appropriate values corresponding to the dimension of $\boldsymbol{\Psi}$, $\boldsymbol{\psi}^j$ is the j-th row of $\boldsymbol{\Psi}$, and $\otimes$ represents the Hadamard product [67]. The $\text{vec}(\cdot)$ operator and Hadamard product are also explained in Appendix A.

Liu et al. [66] defined the variance-detecting sensitivity as the ratio of the change in the variance of measurements over a perturbation of the variance components in

$\boldsymbol{\sigma}$, i.e.,

$$S \equiv \min_{\delta\boldsymbol{\sigma}\neq 0} \frac{\mathrm{tr}((\delta\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}})^T \delta\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}})}{(\delta\boldsymbol{\sigma})^T(\delta\boldsymbol{\sigma})}$$

where δ denotes the perturbation operator and $\mathrm{tr}(\cdot)$ denotes the trace of a matrix. It is shown in Appendix B that the sensitivity can be equivalently written as

$$S = \lambda_{\min}\left(\boldsymbol{\pi}(\boldsymbol{\Psi})^T \boldsymbol{\pi}(\boldsymbol{\Psi})\right)$$

where $\lambda_{\min}(\cdot)$ is the smallest eigenvalue of a matrix.

It is now clear that the variance-detecting sensitivity actually belongs to the family of the so-called alphabetic optimality criteria in the optimal experiment design, including A-optimality, D-optimality, and E-optimality, which were defined using an algebraic form of the eigenvalues of the Fisher information matrix [68]. Here, $\boldsymbol{\pi}(\boldsymbol{\Psi})^T \boldsymbol{\pi}(\boldsymbol{\Psi})$ is actually the Fisher information matrix for detecting the variance components and this index S is in fact the E-optimality criterion. Liu et al. [66] further stated that maximizing S is also equivalent to minimizing the maximum variance of a linear parametric function of the variation components to be estimated. The detailed model development can be found in [66] and [69].

To formulate this optimization problem, denote by X_i, Y_i, and Z_i the coordinates of where the i-th sensor is located, and by Q_i the station on which the i-th sensor is placed. Then, the sensor placement can be represented by $\omega = [X_1\ Y_1\ Z_1\ Q_1\ \ldots\ X_d\ Y_d\ Z_d\ Q_d]$. The geometric constraint (since a sensor can only measure some valid area on a product) is represented by $G(\omega) \geq 0$. As such, the sensor placement problem for a specified number of sensors is to find the optimal sensor locations that maximize the sensitivity S, namely:

$$\max_{\omega} S(\omega) = \lambda_{\min}\left(\boldsymbol{\pi}(\boldsymbol{\Psi})^T \boldsymbol{\pi}(\boldsymbol{\Psi})\right) \quad \text{subject to } G(\omega) \geq 0, \tag{5.6}$$

Note that this formulation is the same as (1.1).

To facilitate the application of the AEMC algorithm, we discretize the geometric area of each part viable for sensor placement using a resolution of 10 mm (which is the size of a locator's diameter); this treatment is the same as what was done in [6] and [66]. The discretization procedure also ensures that all physical constraints are incorporated into the candidate samples so that we can solve the unconstrained optimization (1.2) for sensor placement. This discretization results in N_c candidate locations for any single sensor; hence the sample space for (1.2) is $\mathcal{X} = [1, N_c]^d \cap \mathbf{Z}^d$.

One more detail to note is that for the sensor placement problem, we want to *maximize* the sensitivity objective function (i.e., the more sensitive a sensor system is, the better), while AEMC is to solve a minimization problem. For this reason, we let $H(x)$ in the AEMC algorithm be equal to the sensitivity response of x multiplied by -1, where x represents an instance of sensor placement.

2. Three-station Example

In this section, we attempt to find an optimal sensor placement strategy in the three-station assembly process in Fig. 9. For this assembly process, the 10-mm resolution level results in the number of candidate sensor locations on each part as $n_1 = 6,650, n_2 = 7,480, n_3 = 2,600$, and $n_4 = 2,600$. Because part 1 and 2 appear on all three stations and part 3 and 4 appear on the second and third stations, there are a total of $N_c = 3 \times (n_1 + n_2) + 2 \times (n_3 + n_4) = 52,790$ candidate locations for each sensor. Suppose that $d = 9$, meaning that nine sensors are to be installed, then the total number of solution candidates is $C_9^{52,790} \approx 8.8 \times 10^{36}$, where C_a^b is a combinational operator. Obviously the number of solution candidates is overwhelmingly large. In this section, we solve the sensor placement problem for nine sensors and 20 sensors, respectively.

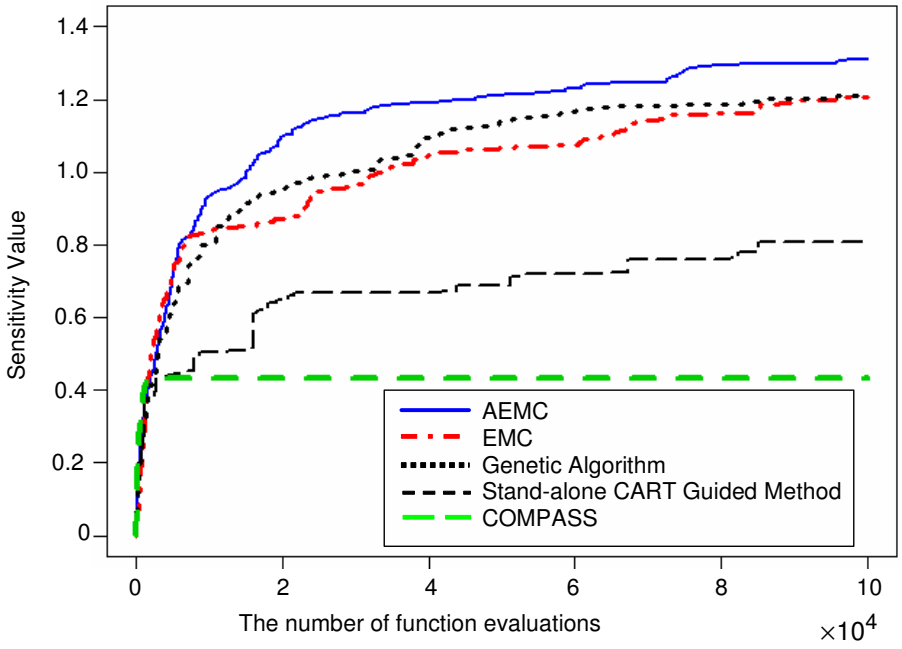


Fig. 10. Algorithm performances for nine sensors on three stations

For the scenario of $d = 9$, each algorithm is run for 10^5 function evaluations. The results of various methods are presented in Fig. 10. It demonstrates a clear advantage of AEMC over the other algorithms. EMC and genetic algorithm have similar performances. After about 4×10^4 function evaluations, AEMC finds $H(x) \approx 1.20$, which is the best value found by EMC and genetic algorithm after 10^5 function evaluations. This translates to 2.5 times improvement in terms of CPU time.

Fig. 11 gives a Boxplot of the best sensitivity values found at the end of each algorithm. AMEC finds a sensitivity value, on average, 10% better than EMC and genetic algorithm. AEMC also has smaller uncertainty than EMC. From the two figures, it is worth noting that the stand-alone CART guided method performs much worse than the other algorithms in this example. We believe this happens mainly because the stand-alone CART guided method fails to gather representative data in the sample space associated with the problem. Also, as can be seen, COMPASS is

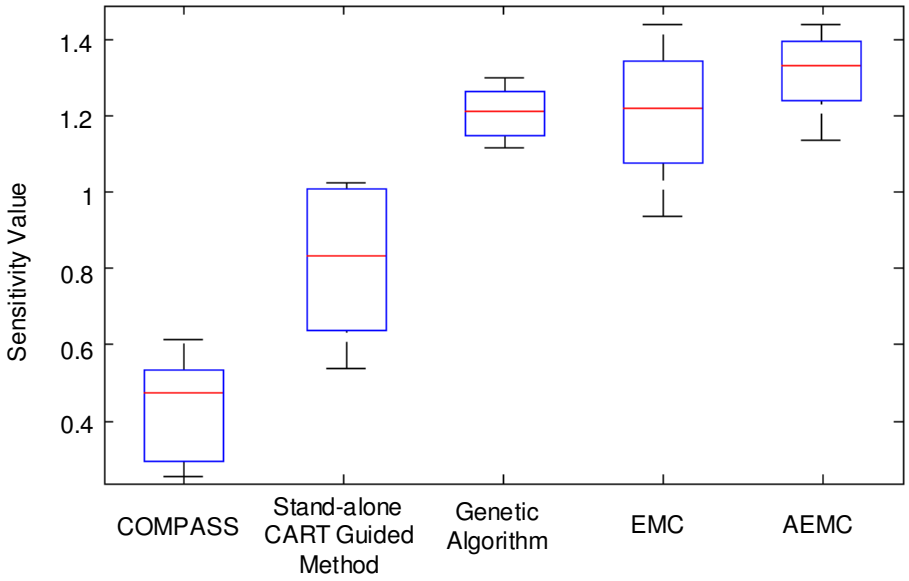


Fig. 11. Uncertainty of different algorithms for nine sensors on three stations

trapped into local optima at an early stage and can not improve further.

Fig. 12 presents the best (i.e., yielding the largest sensitivity) sensor placement strategy found in this example.

We also test the AEMC method in a higher dimensional case, i.e., when $d = 20$. All the algorithms are again run for 10^5 function evaluations. The algorithm performance curves are presented in Fig. 13, where we observe that the sensitivity value found by AEMC after 3×10^4 function evaluations is the same as that found

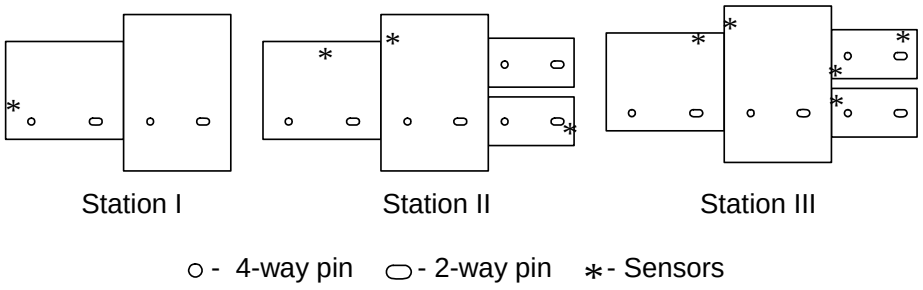


Fig. 12. Best sensor placement for nine sensors on three stations

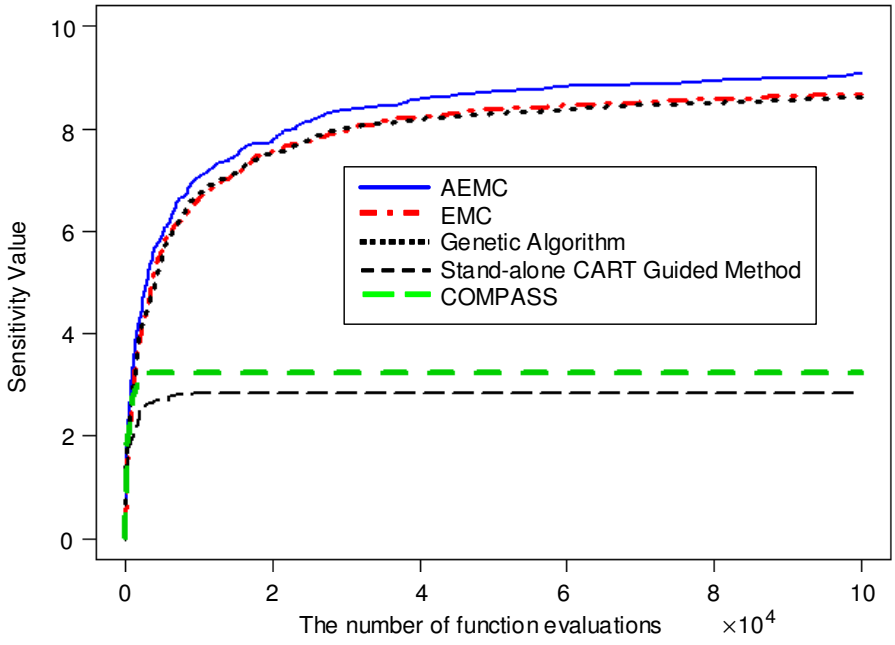


Fig. 13. Algorithm performances for 20 sensors on three stations

by EMC or genetic algorithm after 10^5 function evaluations. This translates to a 3-fold improvement in terms of CPU time. Interestingly, this improvement is greater than that in the 9-sensor case. The final sensitivity value attained by AMEC is, on average, 7% better than EMC and genetic algorithm. Again, the stand-alone CART guided method and COMPASS fail to compete with the other algorithms. We feel that as the dimensionality of the sample space gets higher, the performance of the stand-alone CART guided method gets worse compared to others. Fig. 14 shows the uncertainty of each algorithm. In this case, AEMC has a little higher uncertainty than EMC, but the average results of AEMC are still better. Fig. 15 presents the best sensor placement strategy found in this example.

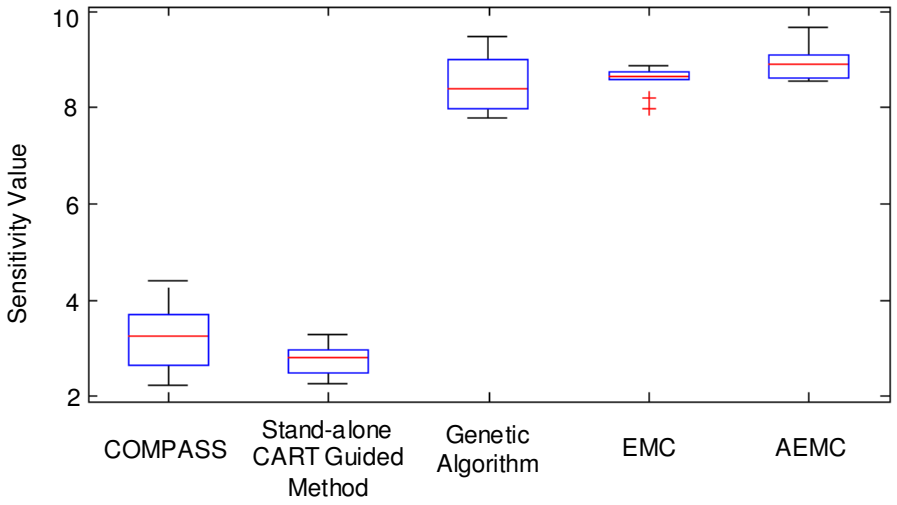


Fig. 14. Uncertainty of different algorithms for 20 sensors on three stations

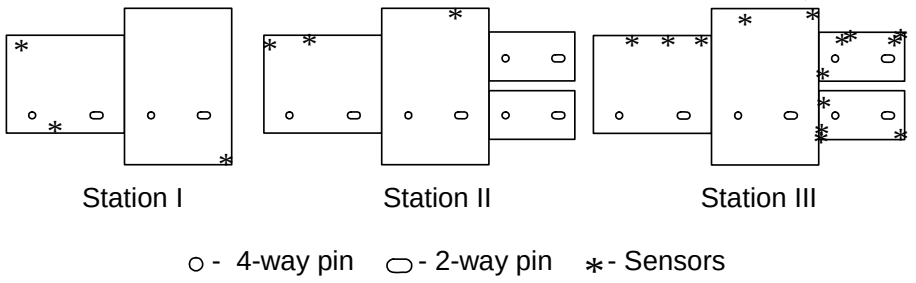


Fig. 15. Best sensor placement for 20 sensors on three stations

3. Six-station Assembly Process Example

Now we solve a sensor placement problem in a more complex assembly process, a six-station assembly process shown in Fig. 16. As can be seen, six parts are assembled on six stations. This example is more difficult because the sample space is even larger than that for the three-station example. For this assembly process, the 10-mm resolution level results in the number of candidate sensor locations on each part as $n_1 = 6,650$, $n_2 = 7,480$, $n_3 = 2,600$, $n_4 = 6,650$, $n_5 = 7,480$, and $n_6 = 2,600$. Similar to the three-station example, we can calculate that there are a total of $N_c = 6 \times (n_1 + n_2) + 5 \times n_3 + 4 \times n_4 + 3 \times n_5 + 2 \times n_6 = 152,020$ candidate locations for each sensor. In this section, we solve the optimal sensor placement problem for 20 sensors. Then the total number of solution candidates is $C_{20}^{152,020} \approx 1.8 \times 10^{85}$.

The performance curves of different algorithms are shown in Fig. 17. As can be seen, the sensitivity value found by AEMC after 3×10^4 function evaluations is the same as that found by EMC or genetic algorithm after 10^5 function evaluations. This again translates to a 3-fold improvement in terms of CPU time. The final sensitivity value attained by AMEC is, on average, 11% better than EMC and genetic algorithm. Again, the stand-alone CART guided method and the COMPASS algorithm perform much worse than the other algorithms, mainly because of complexity of the sample space. Fig. 18 shows the uncertainty of each algorithm. The performance of AEMC has relatively smaller uncertainty than the other algorithms. Fig. 19 presents the best sensor placement strategy found in this example.

B. Test Functions

In order to show the potential applicability of AEMC to other optimization problems, we test it on a suite of well-known test functions for global optimization. These test

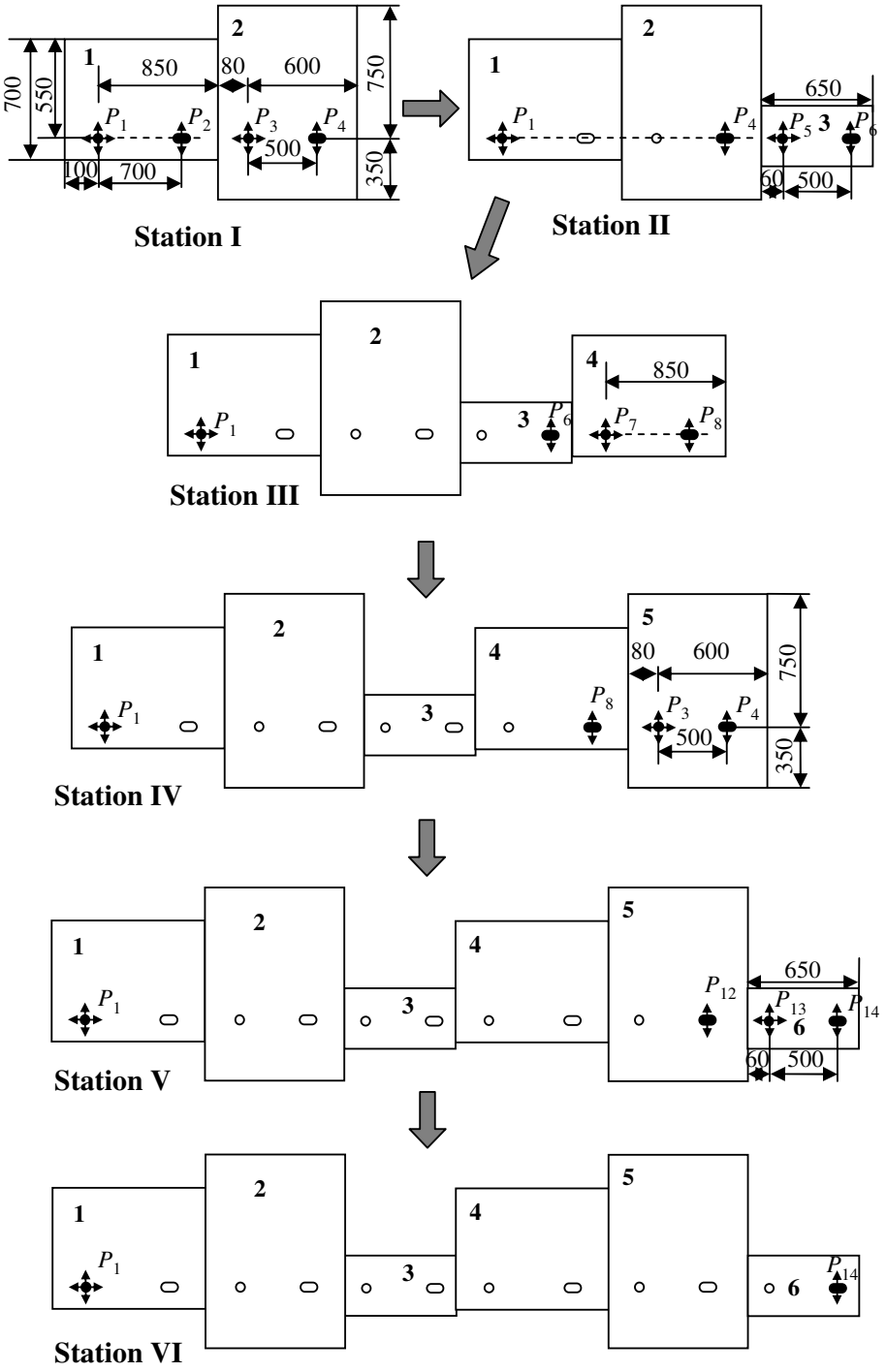


Fig. 16. A six-station assembly process

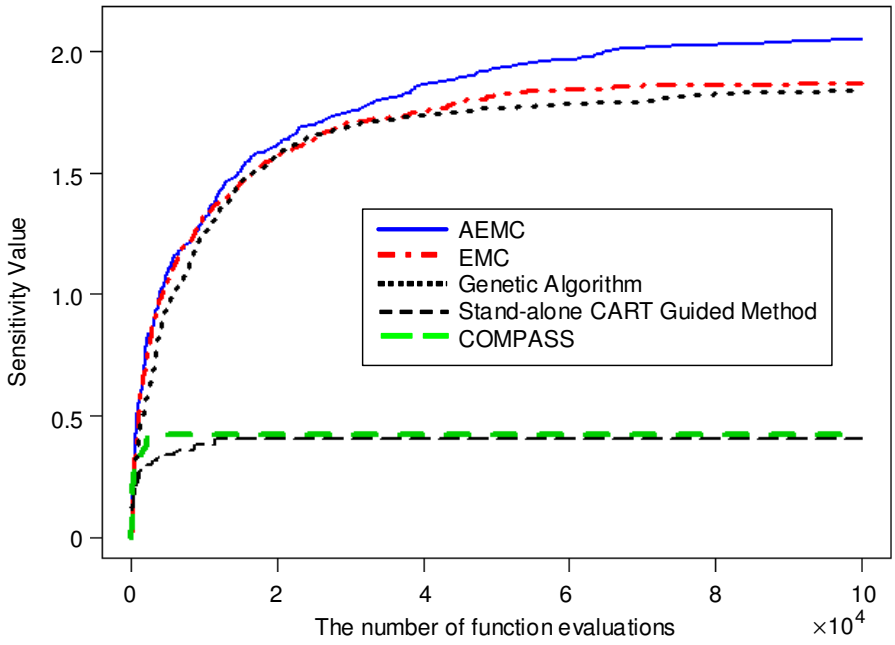


Fig. 17. Algorithm performances for 20 sensors on six stations

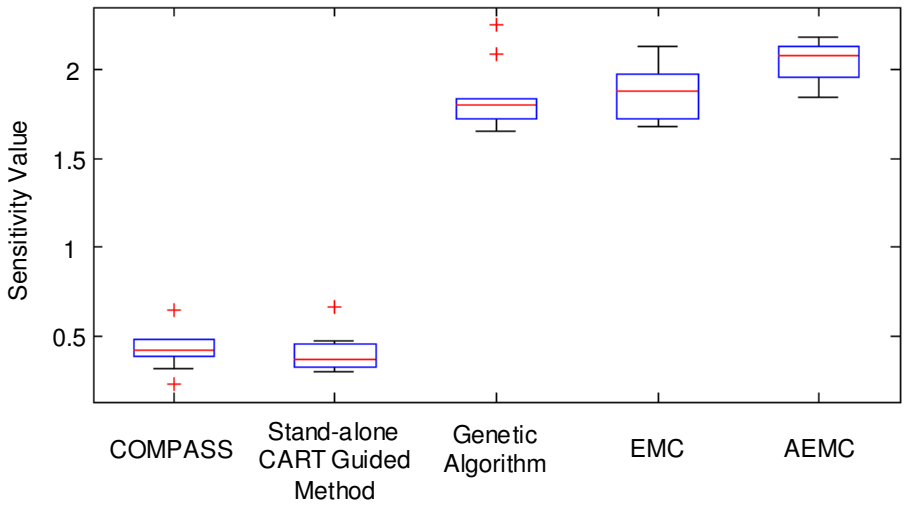


Fig. 18. Uncertainty of different algorithms for 20 sensors on six stations

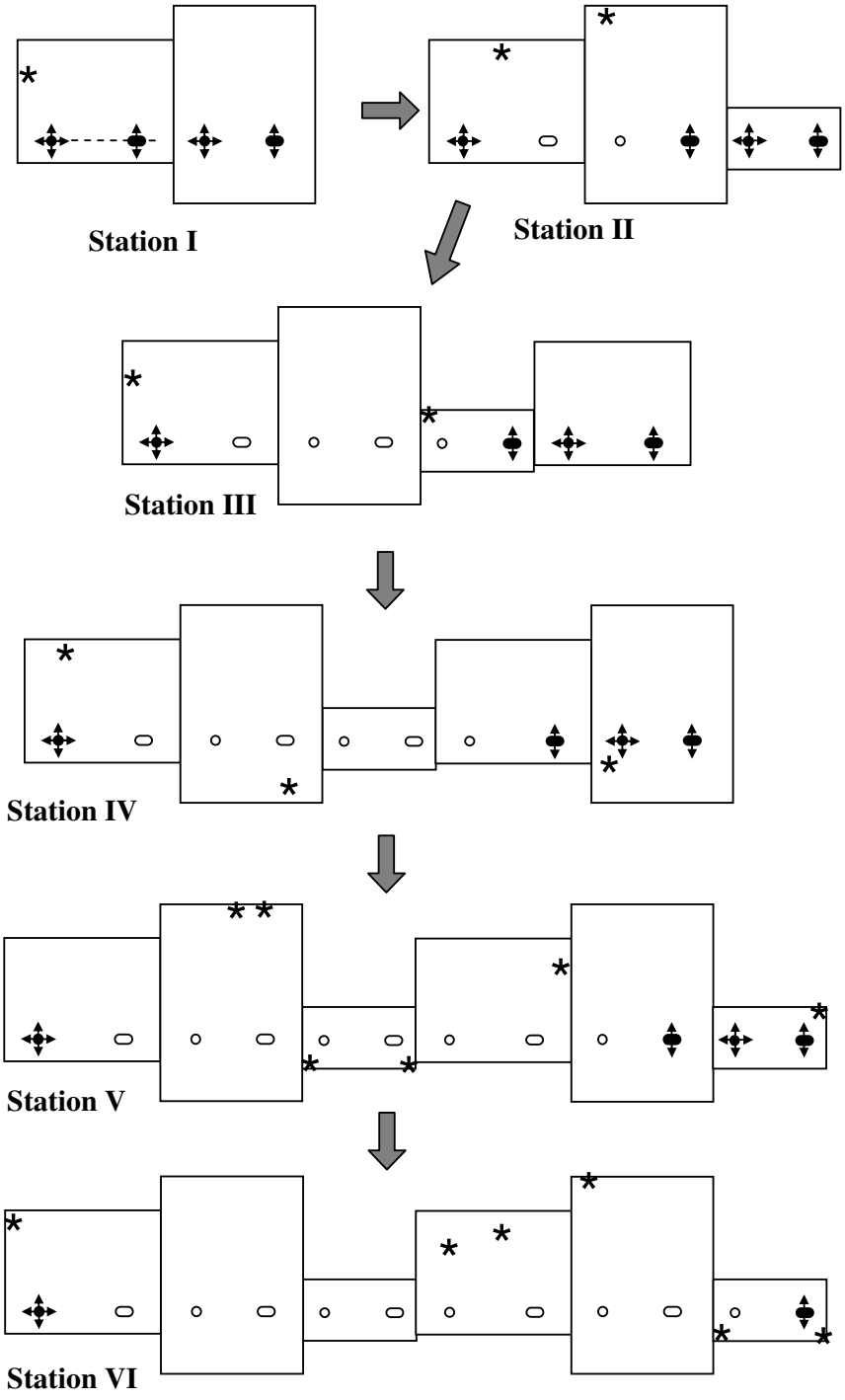


Fig. 19. Best sensor placement for 20 sensors on six stations

functions have multiple local optima and thus are difficult to optimize. We test AEMC in both low and high dimensional sample spaces.

The following test functions, which are commonly used to test global optimization algorithms [57]–[59], are used in our experiments.

(1) Shekel's Foxhole function ($d = 2$),

$$H_S(x) = -\sum_{j=1}^{30} \frac{1}{\sum_{i=1}^{d}(x_i - d_{ji})^2 + c_j}, \quad 0 \le x_i \le 10.$$

The coefficients in the function $D = \{d_{ji}\}$ and $\overrightarrow{c} = \{c_j\}$ are given in Fig. 20. Here we set $d = 2$. The optimal solution is $x^* = (8.024, 9.147)$, and the minimal of $H_S(x^*) = -12.119$.

(2) Rastrigin's function ($d = 20, 50$),

$$H_R(x) = 10d + \sum_{i=1}^{d} \left(x_i^2 - 10\cos(2\pi x_i)\right), \quad -5.12 \le x_i \le 5.12.$$

The global minimum is located at the origin and the global minimum value is 0. We test AEMC in two cases: $d = 20$ and $d = 50$.

(3) Griewank function ($d = 50$),

$$H_G(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos(\frac{x_i}{\sqrt{i}}) + 1,$$

$$-600 \le x_i \le 600, \ i = 1, \ldots, d.$$

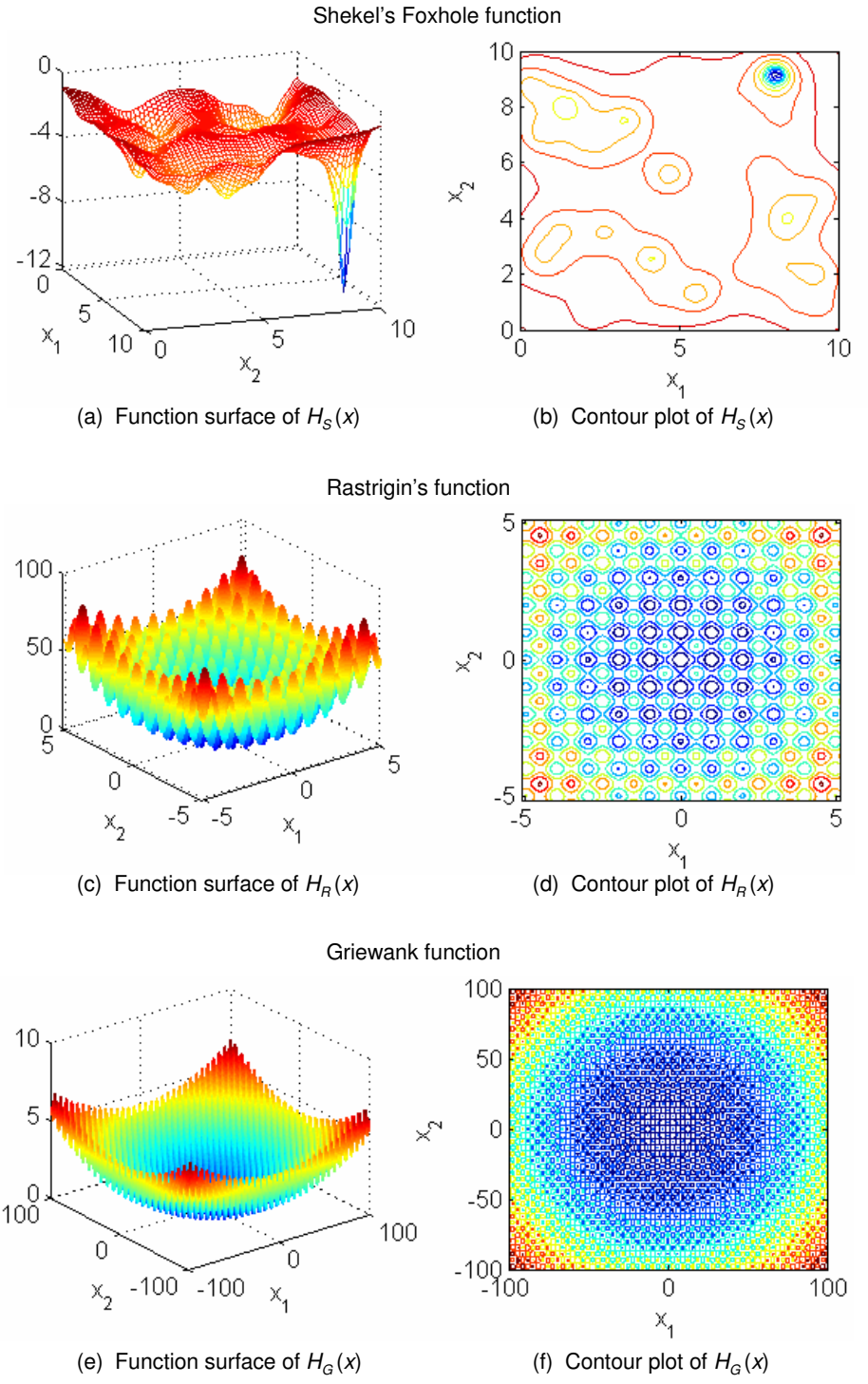
The global minimum is located at the origin and the global minimum value is 0. Here we set $d = 50$.

Fig. 21 shows these functions in two dimensions. The left column shows surfaces of these functions and the right column shows contour plots of corresponding functions. Function H_S is a low dimensional function and has only a few local optima. However, the local optima are separated and relatively far apart. Only a small re-

$$D = \begin{pmatrix}
9.681 & 0.667 & 4.783 & 9.095 & 3.517 & 9.325 & 6.544 & 0.211 & 5.122 & 2.020 \\
9.400 & 2.041 & 3.788 & 7.931 & 2.882 & 2.672 & 3.568 & 1.284 & 7.033 & 7.374 \\
8.025 & 9.152 & 5.114 & 7.621 & 4.564 & 4.711 & 2.996 & 6.126 & 0.734 & 4.982 \\
2.196 & 0.415 & 5.649 & 6.979 & 9.510 & 9.166 & 6.304 & 6.054 & 9.377 & 1.426 \\
8.074 & 8.777 & 3.467 & 1.863 & 6.708 & 6.349 & 4.534 & 0.276 & 7.633 & 1.567 \\
7.650 & 5.658 & 0.720 & 2.764 & 3.278 & 5.283 & 7.474 & 6.274 & 1.409 & 8.208 \\
1.256 & 3.605 & 8.623 & 6.905 & 0.584 & 8.133 & 6.071 & 6.888 & 4.187 & 5.448 \\
8.314 & 2.261 & 4.224 & 1.781 & 4.124 & 0.932 & 8.129 & 8.658 & 1.208 & 5.762 \\
0.226 & 8.858 & 1.420 & 0.945 & 1.622 & 4.698 & 6.228 & 9.096 & 0.972 & 7.637 \\
7.305 & 2.228 & 1.242 & 5.928 & 9.133 & 1.826 & 4.060 & 5.204 & 8.713 & 8.247 \\
0.652 & 7.027 & 0.508 & 4.876 & 8.807 & 4.632 & 5.808 & 6.937 & 3.291 & 7.016 \\
2.699 & 3.516 & 5.874 & 4.119 & 4.461 & 7.496 & 8.817 & 0.690 & 6.593 & 9.789 \\
8.327 & 3.897 & 2.017 & 9.570 & 9.825 & 1.150 & 1.395 & 3.885 & 6.354 & 0.109 \\
2.132 & 7.006 & 7.136 & 2.641 & 1.882 & 5.943 & 7.273 & 7.691 & 2.880 & 0.564 \\
4.707 & 5.579 & 4.080 & 0.581 & 9.698 & 8.542 & 8.077 & 8.515 & 9.231 & 4.670 \\
8.304 & 7.559 & 8.567 & 0.322 & 7.128 & 8.392 & 1.472 & 8.524 & 2.277 & 7.826 \\
8.632 & 4.409 & 4.832 & 5.768 & 7.050 & 6.715 & 1.711 & 4.323 & 4.405 & 4.591 \\
4.887 & 9.112 & 0.170 & 8.967 & 9.693 & 9.867 & 7.508 & 7.770 & 8.382 & 6.740 \\
2.440 & 6.686 & 4.299 & 1.007 & 7.008 & 1.427 & 9.398 & 8.480 & 9.950 & 1.675 \\
6.306 & 8.583 & 6.084 & 1.138 & 4.350 & 3.134 & 7.853 & 6.061 & 7.457 & 2.258 \\
0.652 & 2.343 & 1.370 & 0.821 & 1.310 & 1.063 & 0.689 & 8.819 & 8.833 & 9.070 \\
5.558 & 1.272 & 5.756 & 9.857 & 2.279 & 2.764 & 1.284 & 1.677 & 1.244 & 1.234 \\
3.352 & 7.549 & 9.817 & 9.437 & 8.687 & 4.167 & 2.570 & 6.540 & 0.228 & 0.027 \\
8.798 & 0.880 & 2.370 & 0.168 & 1.701 & 3.680 & 1.231 & 2.390 & 2.499 & 0.064 \\
1.460 & 8.057 & 1.336 & 7.217 & 7.914 & 3.615 & 9.981 & 9.198 & 5.292 & 1.224 \\
0.432 & 8.645 & 8.774 & 0.249 & 8.081 & 7.461 & 4.416 & 0.652 & 4.002 & 4.644 \\
0.679 & 2.800 & 5.523 & 3.049 & 2.968 & 7.225 & 6.730 & 4.199 & 9.614 & 9.229 \\
4.263 & 1.074 & 7.286 & 5.599 & 8.291 & 5.200 & 9.214 & 8.272 & 4.398 & 4.506 \\
9.496 & 4.830 & 3.150 & 8.270 & 5.079 & 1.231 & 5.731 & 9.494 & 1.883 & 9.732 \\
4.138 & 2.562 & 2.532 & 9.661 & 5.611 & 5.500 & 6.886 & 2.341 & 9.699 & 6.500
\end{pmatrix}$$

$$\overrightarrow{c} = \begin{pmatrix}
0.806 & 0.517 & 0.100 & 0.908 & 0.965 & 0.669 & 0.524 & 0.902 & 0.531 & 0.876 & 0.462 \\
0.491 & 0.463 & 0.714 & 0.352 & 0.869 & 0.813 & 0.811 & 0.828 & 0.964 & 0.789 & 0.360 \\
0.369 & 0.992 & 0.332 & 0.817 & 0.632 & 0.883 & 0.608 & 0.326 &
\end{pmatrix}$$

Fig. 20. Coefficients in the Shekel's Foxhole function



(a) Function surface of $H_S(x)$ (b) Contour plot of $H_S(x)$

(c) Function surface of $H_R(x)$ (d) Contour plot of $H_R(x)$

(e) Function surface of $H_G(x)$ (f) Contour plot of $H_G(x)$

Fig. 21. Graphical representation of the test functions

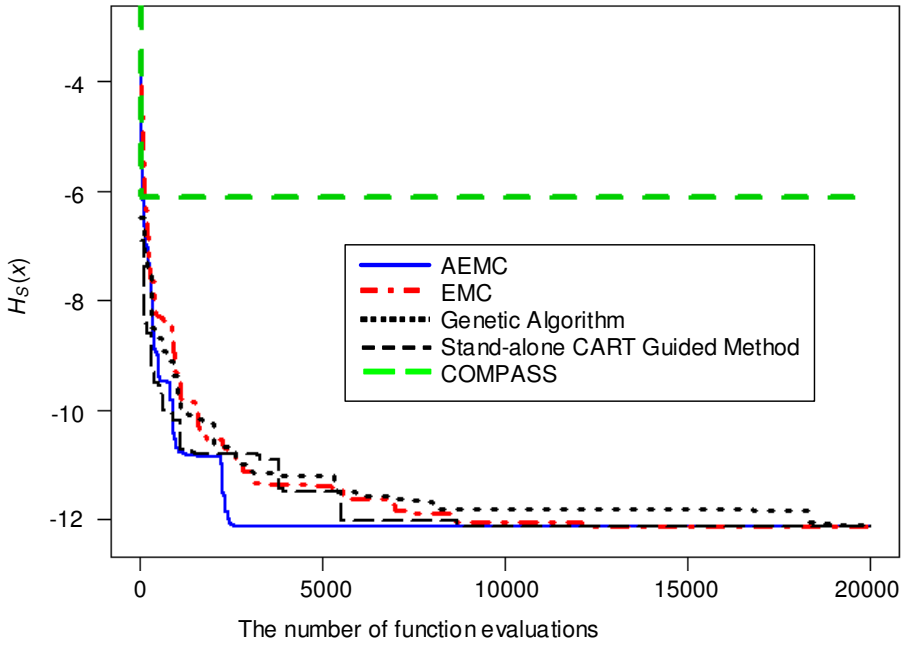


Fig. 22. Algorithm performances for Shekel's Foxhole function

gion contains relatively good solutions. Function H_R is a highly multimodal function. Function H_G is both highly multimodal and skewed, and the number of local minima increases exponentially with the dimension d.

Fig. 22 presents the algorithm performances for function H_S. All algorithms were run for 2×10^4 function evaluations. AEMC clearly outperforms the other algorithms as AEMC converges to the global optimum much faster. As explained earlier, this fast convergence is an appealing property to engineering design problems. In this example, the stand-alone CART guided method, genetic algorithm, and EMC have comparable performances. It is not surprising that the performance of COMPASS is worse than others. Since COMPASS only samples around the current best solution, it is difficult, if not impossible, to jump out of a local optimum.

For function H_R ($d = 20$), the algorithm performances are shown in Fig. 23. All algorithms were run for 5×10^4 function evaluations. Please note that we have trun-

cated the y-axis to be between 0 and 220 so as to show the early-stage performances of different algorithms more clearly. On the bottom panel of Fig. 23, we enlarge the performance curves for a clearer view. The performance curves of AEMC, EMC, and genetic algorithm are relatively close. But AEMC still achieves 2-fold improvement in terms of CPU time comparing to EMC and genetic algorithm, i.e., the objective function value found by AEMC after about 2.5×10^4 function evaluations is the same as that found by EMC/genetic algorithm after 5×10^4 function evaluations. In this example, the stand-alone CART guided method converges much slower than the other algorithms. Again, COMPASS fails to escape local optima.

For H_R ($d = 50$), Fig. 24 shows the algorithm performances. All algorithms were run for 10^5 function evaluations. We have truncated the y-axis to be between 0 and 660 so as to show the early-stage performances of different algorithms more clearly. As the dimension of the sample space increases, the advantage of AEMC becomes more significant. AEMC achieves 2-fold improvement in terms of CPU time compared to EMC and 7-fold improvement compared to genetic algorithm. The performances of the stand-alone CART guided method and COMPASS are much worse than the other algorithms.

Fig. 25 presents the algorithm performances for function H_G ($d = 50$). All algorithms were run for 10^5 function evaluations. We have truncated the y-axis to be between 0 and 400 so as to show the early-stage performances of different algorithms. AEMC clearly outperforms the other algorithms, especially in the beginning stage, as one can observe that AEMC converges much faster. In this example, the stand-alone CART guided method converges faster than the standard genetic algorithm, COMPASS, and EMC, but is entrapped into a local optimum at an early stage. AEMC appears to level off after 65,000 function evaluations. However, as assured by Theorem 2 in Chapter IV, AEMC will eventually reach the global optimum if given

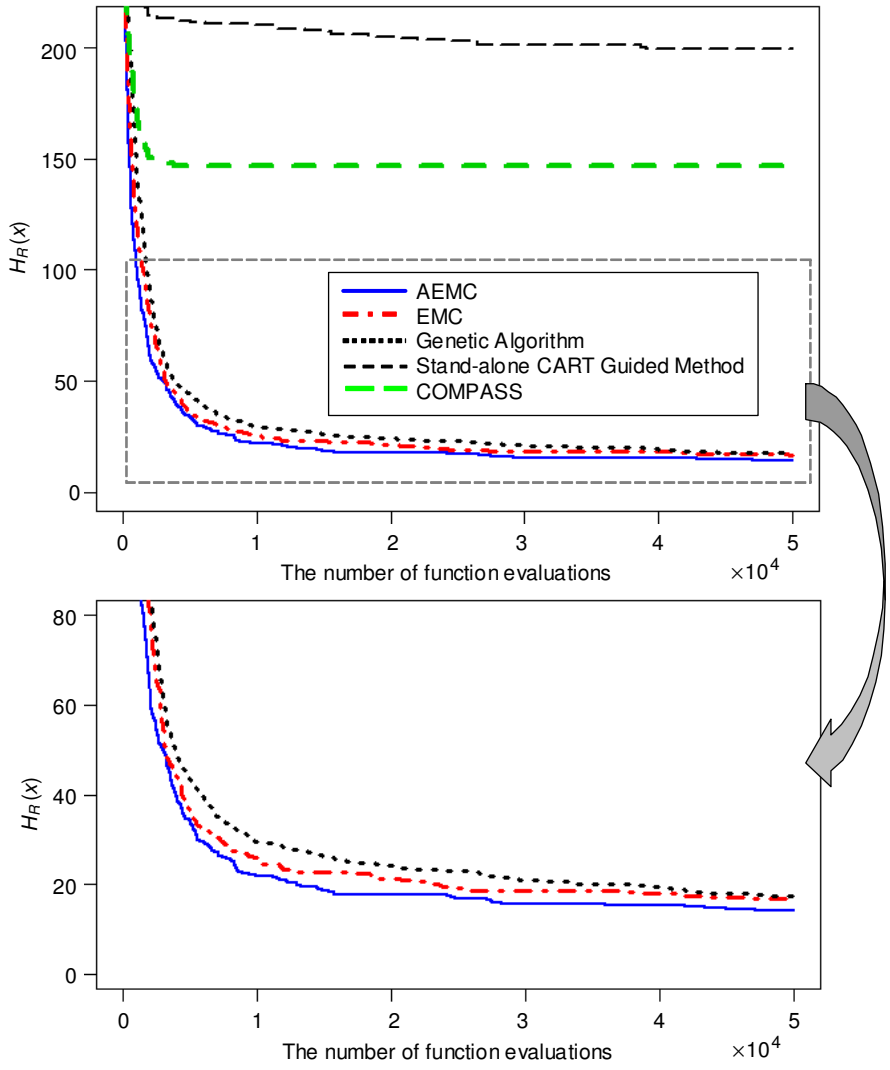


Fig. 23. Algorithm performances for Rastrigin's function ($d = 20$)

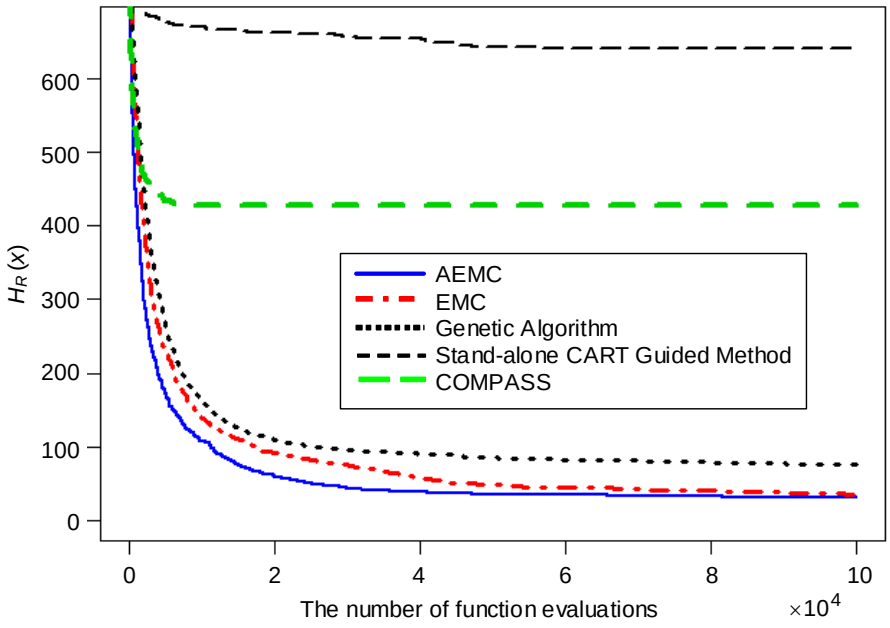


Fig. 24. Algorithm performances for Rastrigin's function ($d = 50$)

enough computational effort.

For each test case, Fig. 26 gives a Boxplot of the best function values found at the end of the algorithms. Comparing to the other methods, the AEMC algorithm not only improves the average performance but also reduces the uncertainty. Since the algorithms are run for a large number of function evaluations, genetic algorithm and EMC also find good solutions at the end of the algorithms. However, as shown in the performance curve figures (Fig. 22 $\sim$ 25), AEMC finds good solutions much faster.

C. Sensitivity Analysis

We run an ANOVA analysis to investigate how sensitive the performance of AEMC to the tuning parameters: the switching condition M, the percentile value h, and the tree size J_H. This analysis does not intend to provide a set of universal rules for

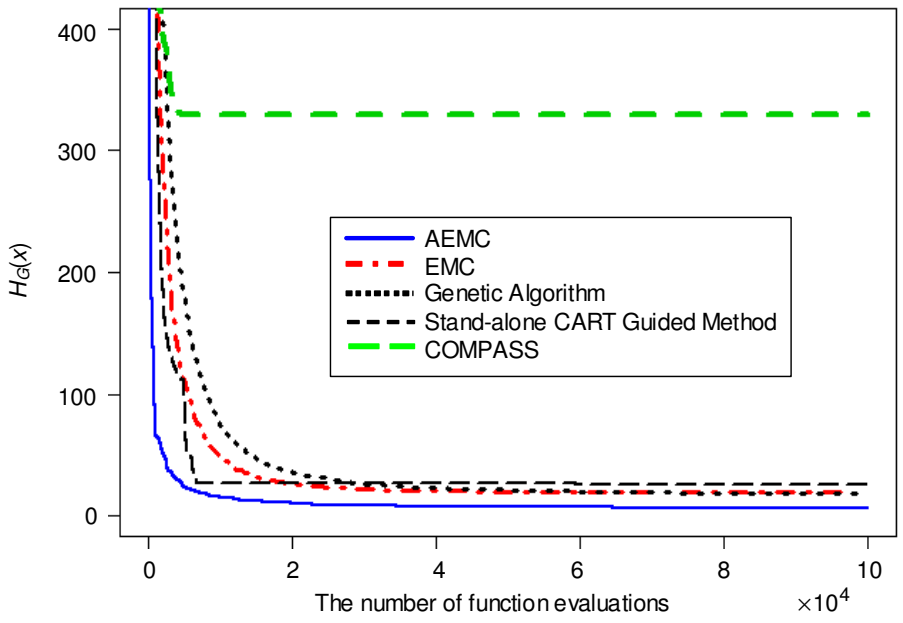


Fig. 25. Algorithm performances for Griewank function

choosing appropriate M, h, and J_H values, because specific choices of the parameter values should depend on the complexity of the problem and the characteristics of the objective function. The following results should be considered as a general understanding about the effects of the tuning parameters. Users could follow our recommendations outlined in Section C.4 in Chapter III and fine tune the algorithm parameters for their problems.

In the ANOVA analysis, the value of M is chosen from five levels (10, 30, 60, 90, 120), the value of h is chosen from three levels (1%, 10%, 20%), and the value of J_H is chosen from three levels (3, 6, 12). Then a full factorial design with 45 cases is constructed. Throughout the following sensitivity analysis, we set $n = 5$, $p_m = .25$. We use both the sensor placement example and test function example in the ANOVA analysis.

In the following analysis, we use two kinds of measures: (1) the objective function

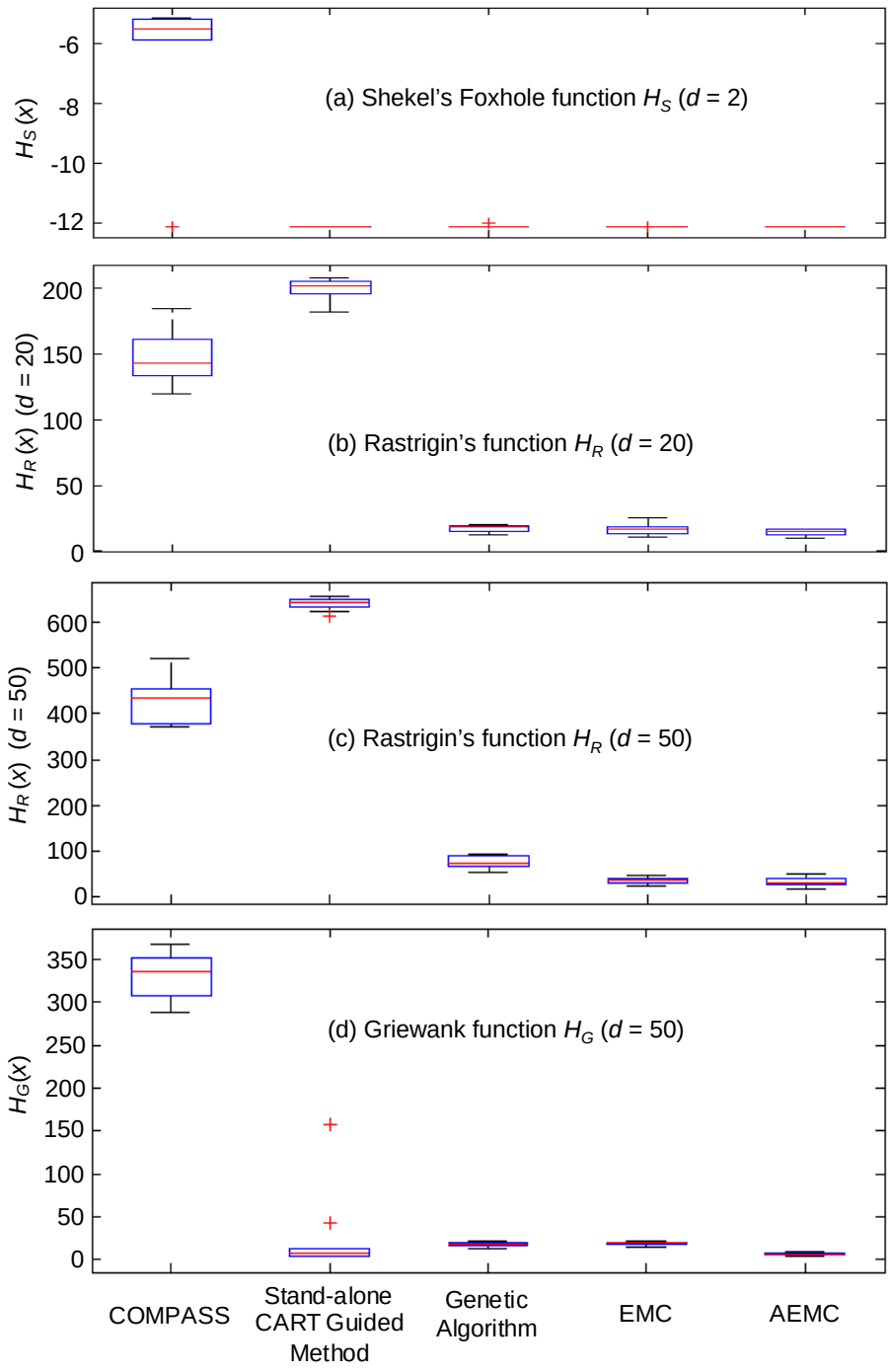


Fig. 26. Uncertainty of different algorithms for the test functions: (a) H_S ($d = 2$); (b) H_R ($d = 20$); (c) H_R ($d = 50$); (d) H_G ($d = 50$)

value achieved after a pre-specified number of function evaluations; (2) the number of function evaluations needed to achieve a pre-specified objective function value. These two measures are used in the previous algorithm comparisons in Section A and B of this chapter. The first measure assesses the capability of AEMC to escape local optima to find a good solution; the second measure assesses the convergence rate of AEMC to a good solution. It is not difficult to see that these two measures are strongly correlated with each other, i.e., if AEMC performs well in terms of one measure, it is likely to perform well in terms of the other measure. Thus, they are expected to generate similar conclusions in the ANOVA analysis (our following numerical results validate this understanding).

We first use the objective function value after a pre-specified number of function evaluations as the response in the ANOVA analysis. For the sensor placement example, we use the 9-sensor case for the ANOVA analysis. AEMC was run for 10^5 function evaluations, and we recorded the best function value found as the output. For each factor level combination, this was done five times. The ANOVA table is shown in Table II. We can see that the main effects of M and h are significant at the .05 level.

Fig. 27 shows the main effect plots for M and h. As can be seen, a relatively smaller M leads to better algorithm performance. The sensitivity values found by AEMC with $M = 10$ and $M = 30$ are better than those with larger M values (note that a larger sensitivity value is preferred in this sensor placement example). However, M should not be too small. As we argued before, if M is too small, EMC may not have gathered enough representative samples yet and thus the data-mining operations will not be effective. The main effect plot validate this argument: the sensitivity value decreases when M changes from 30 to 10. As to the effect of h, we observe that it is critical for the performance of AEMC and that $h = 10\%$ result in better algorithm

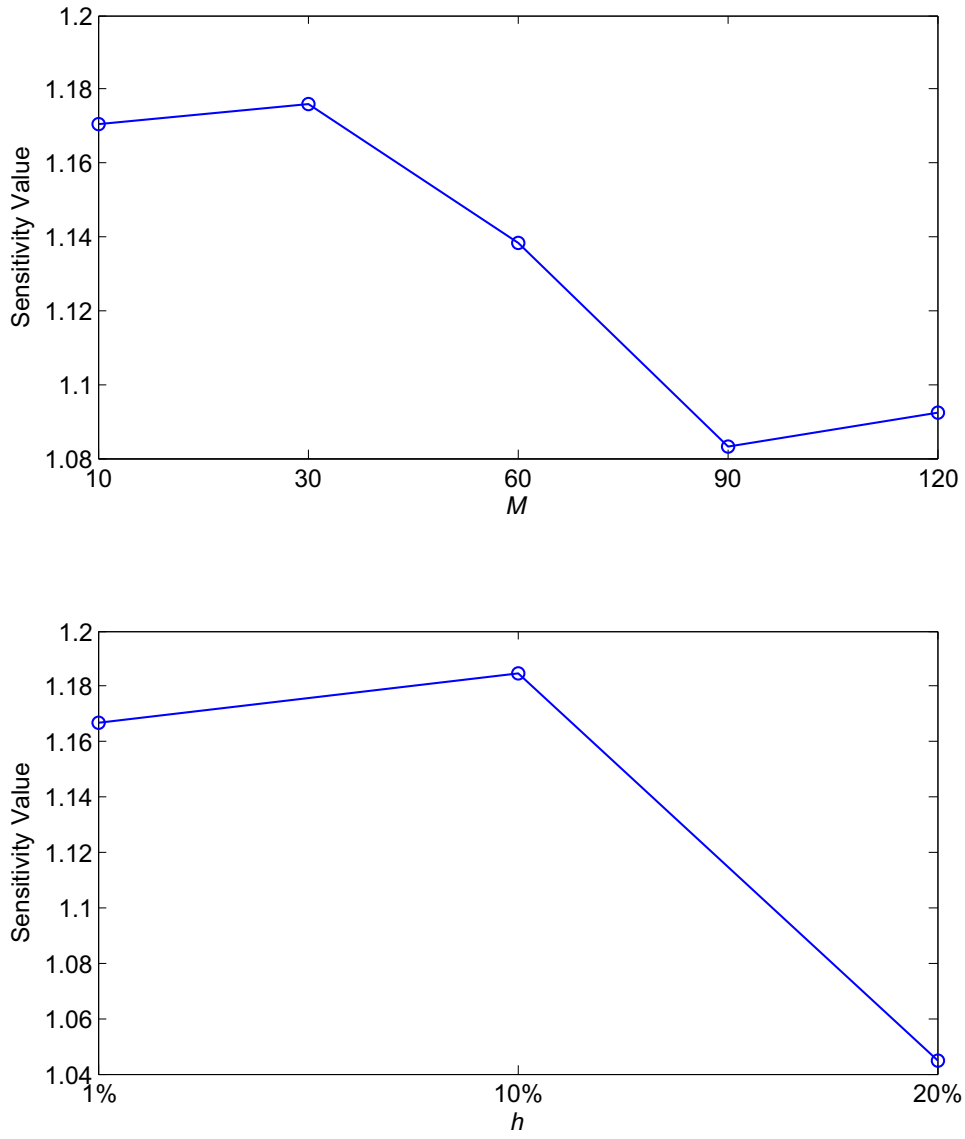


Fig. 27. Main effect plots for M and h for the sensor placement example (using objective function value)

Table II. ANOVA analysis for the sensor placement example (using objective function value)

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
M	0.33	4	0.08	2.43	0.05
h	0.87	2	0.44	12.75	0.00
J_H	0.06	2	0.03	0.84	0.43
$M \times h$	0.26	8	0.03	0.94	0.48
$M \times J_H$	0.14	8	0.02	0.53	0.84
$h \times J_H$	0.11	4	0.03	0.79	0.53
Error	6.70	196	0.03		
Total	8.47	224			

performance than the other two h values. This observation validates our argument in Section C.4 in Chapter III: AEMC with a too small h focuses too much on the peaks of the objective function surface and AEMC with a too large h pays too much attention to unpromising search regions; both result in unsatisfactory performances. In general, relatively smaller M and h are favored.

For the Griewank example, we ran AEMC for 5×10^4 function evaluations and recorded the best function value found as the output. For each factor level combination, this was done five times. The ANOVA table is shown in Table III. If using .05 level, then we can see that the main effects of M and h are significant.

The main effect plots for M and h are shown in Fig. 28. In this example, the performance of AEMC decreases as M increases (in this example, the smaller $H_G(x)$ is, the better). The reason is that data-mining operations are effective for this problem as shown in Fig. 25. Thus switching to the data-mining mode more frequently will lead to better performance. The main effect plot of h looks similar to

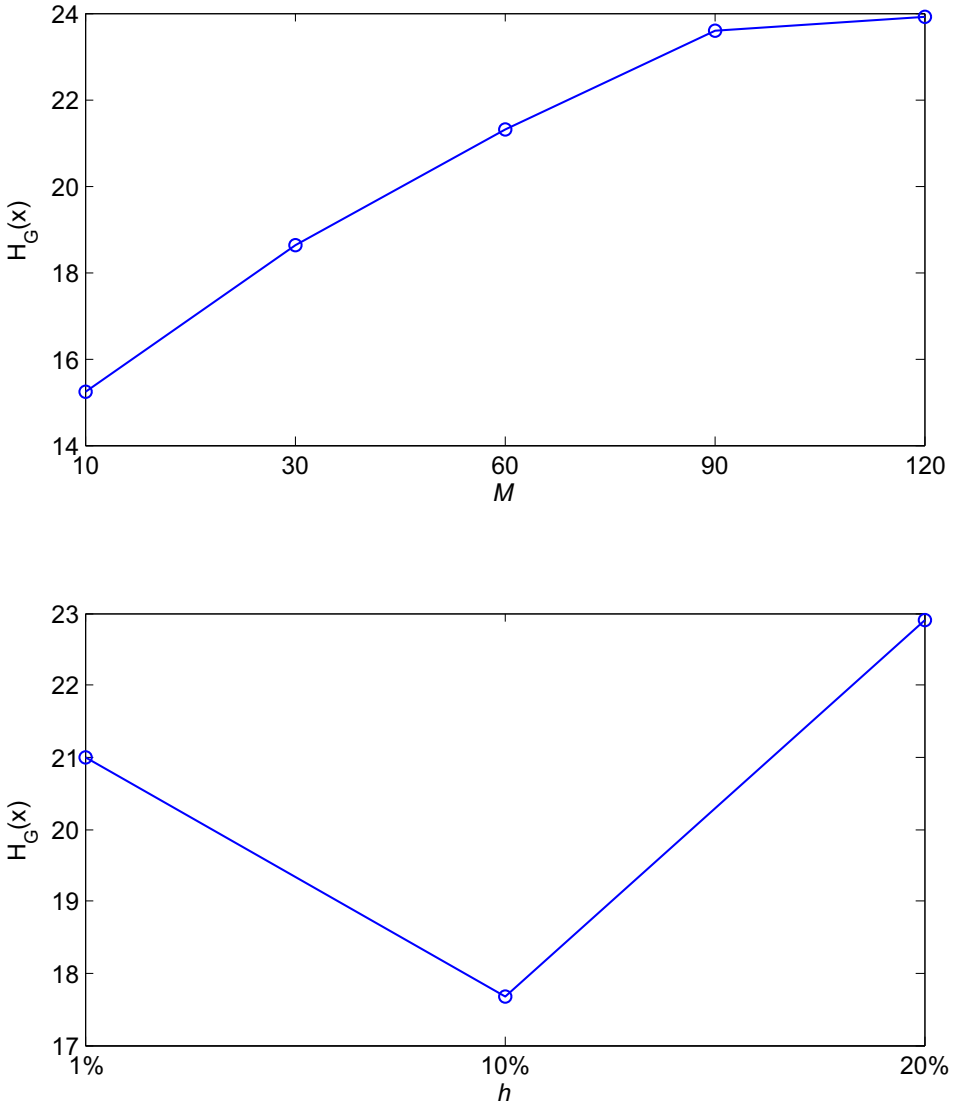


Fig. 28. Main effect plots for M and h for the Griewank function example (using objective function value)

Table III. ANOVA analysis for the Griewank example (using objective function value)

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
M	2393.40	4	598.35	48.50	0.00
h	1046.26	2	523.13	42.40	0.00
J_H	0.18	2	0.09	0.01	0.99
$M \times h$	163.51	8	20.44	1.66	0.11
$M \times J_H$	104.15	8	13.02	1.06	0.40
$h \times J_H$	43.41	4	10.85	0.88	0.48
Error	2418.08	196	12.34		
Total	6169.00	224			

the previous example; $h = 10\%$ gives a better algorithm performance than the other h values.

Now we use the number of function evaluations needed to reach a pre-specified objective function value as the response in the ANOVA analysis. For the sensor placement example, we ran AEMC until it found a objective function value of one and we recorded the number of function evaluations needed as the response. For each factor level combination, this was done five times. The ANOVA table for the sensor placement example is shown in Table IV. Using .05 level, we can see again that the main effects of M and h are significant. The main effect plots are shown in Fig. 29. We observe that the trends in the main effect plots are similar to Fig. 27. Please note that Fig. 27 and Fig. 29 should be compared in a mirrored way. The reason is that a larger value in the vertical axis in Fig. 27 (meaning a larger objective function value) is preferred while a smaller value in the vertical axis in Fig. 29 (meaning faster convergence) is preferred. It is observed that $M = 30$ and $h = 10\%$ work the best for AEMC.

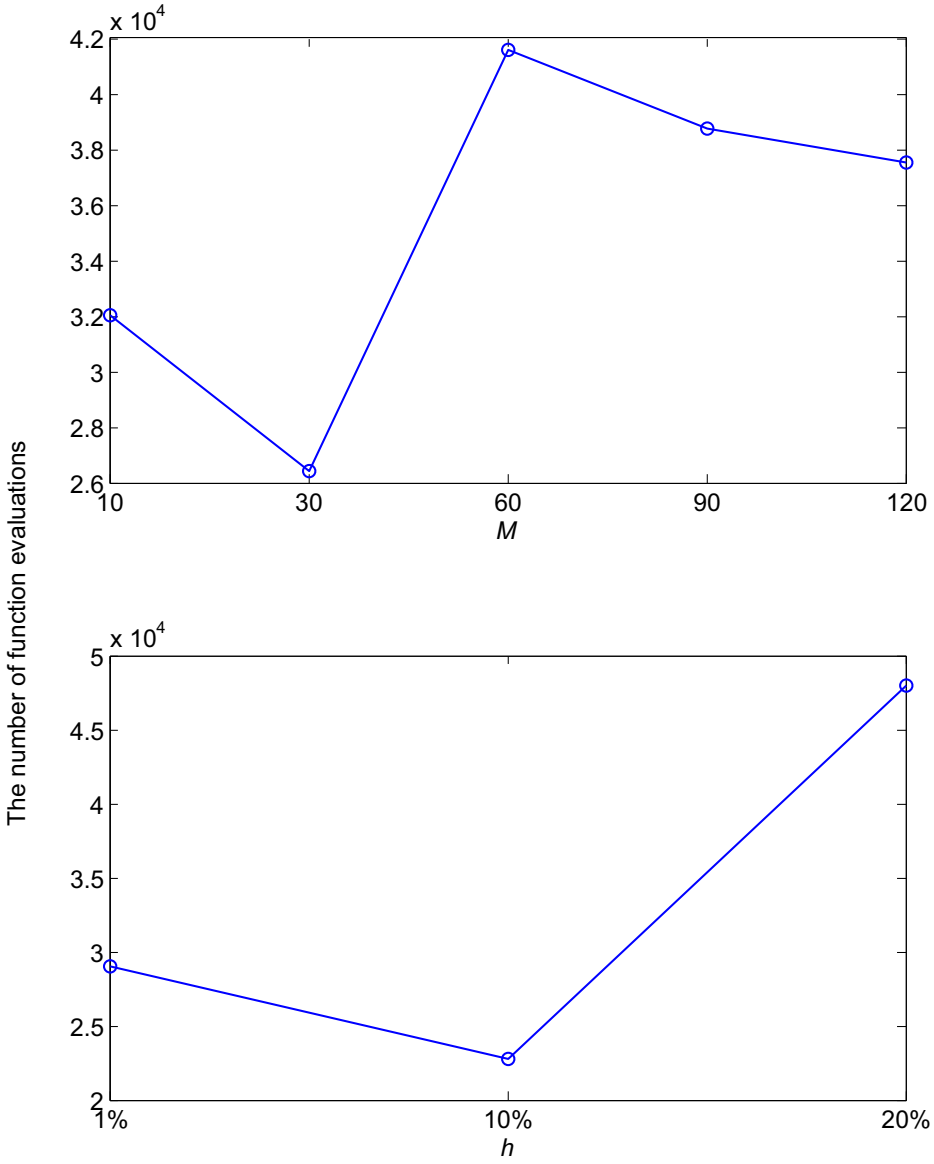


Fig. 29. Main effect plots for M and h for the sensor placement example (using the number of function evaluations)

Table IV. ANOVA analysis for the sensor placement example (using the number of function evaluations)

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
M	8.48e+009	4	2.12e+009	2.8	0.03
h	2.57e+010	2	1.28e+010	16.94	0.00
J_H	2.20e+009	2	1.10e+009	1.45	0.24
$M \times h$	1.60e+010	8	2.00e+009	2.63	0.01
$M \times J_H$	7.00e+009	8	8.75e+008	1.16	0.33
$h \times J_H$	2.29e+009	4	5.73e+008	0.76	0.55
Error	1.49e+011	196	7.58e+008		
Total	2.10e+011	224			

From Table IV, we also observe that the interaction effect between M and h is significant here, while it was not significant in Table II when the objective function value found after a pre-specified number of function evaluations was used as the response. This indicates that the interaction effect is more critical to the number of function evaluations needed to reach a pre-specified objective function value, i.e., it is critical to the convergence rate of AEMC. The interaction effect plot is given in Fig. 30. It can be seen from the figure that the curve associated with $h = 10\%$ is flatter than the other two and that the curve associated with $M = 30$ is flatter than the others. That is, when either the value of M or h is selected appropriately, the performance of AEMC is less dependent on the value of the other. Fig. 30 actually shows that the combinations $(M = 10,\ h = 10\%)$, $(M = 30,\ h = 1\%)$, and $(M = 90,\ h = 10\%)$ result in slightly better performances than the combination $(M = 30,\ h = 10\%)$. However, from a robust design viewpoint (refer to Chapter 10 in [70]), we still prefer $(M = 30,\ h = 10\%)$ because $M = 30$ and $h = 10\%$ gives the

Table V. ANOVA analysis for the Griewank example (using the number of function evaluations)

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
M	9.29e+011	4	2.32e+011	12.66	0.00
h	1.04e+012	2	5.20e+011	28.34	0.00
J_H	6.69e+010	2	3.34e+010	1.82	0.16
$M \times h$	4.68e+011	8	5.85e+010	3.19	0.00
$M \times J_H$	1.69e+011	8	2.11e+010	1.15	0.33
$h \times J_H$	1.57e+011	4	3.92e+010	2.13	0.08
Error	3.60e+012	196	1.83e+010		
Total	6.42e+012	224			

smallest variance in terms of the algorithm performance (i.e., curves are flatter in the figure).

For the Griewank example, we ran AEMC until it found a objective function value of 20 and we recorded the number of function evaluations needed as the response. For each factor level combination, this was done five times. The ANOVA table is shown in Table V. Using .05 level, we can see that the main effects of M and h and the interaction effect between M and h are significant. The main effect plots and interaction effect plot are shown in Fig. 31 and Fig. 32, respectively. Again, the trends in the main effect plots are similar to Fig. 28. From the main effect plots, we see that $M = 10$ and $h = 10\%$ work the best for AEMC. From Fig. 32, it is observed that the curve associated with $h = 10\%$ is flatter than the other two and that the curve associated with $M = 10$ is flatter than the others, which indicates again that if one of the values from M and h is chosen appropriately, the performance of AEMC is less dependent on the value of the other.

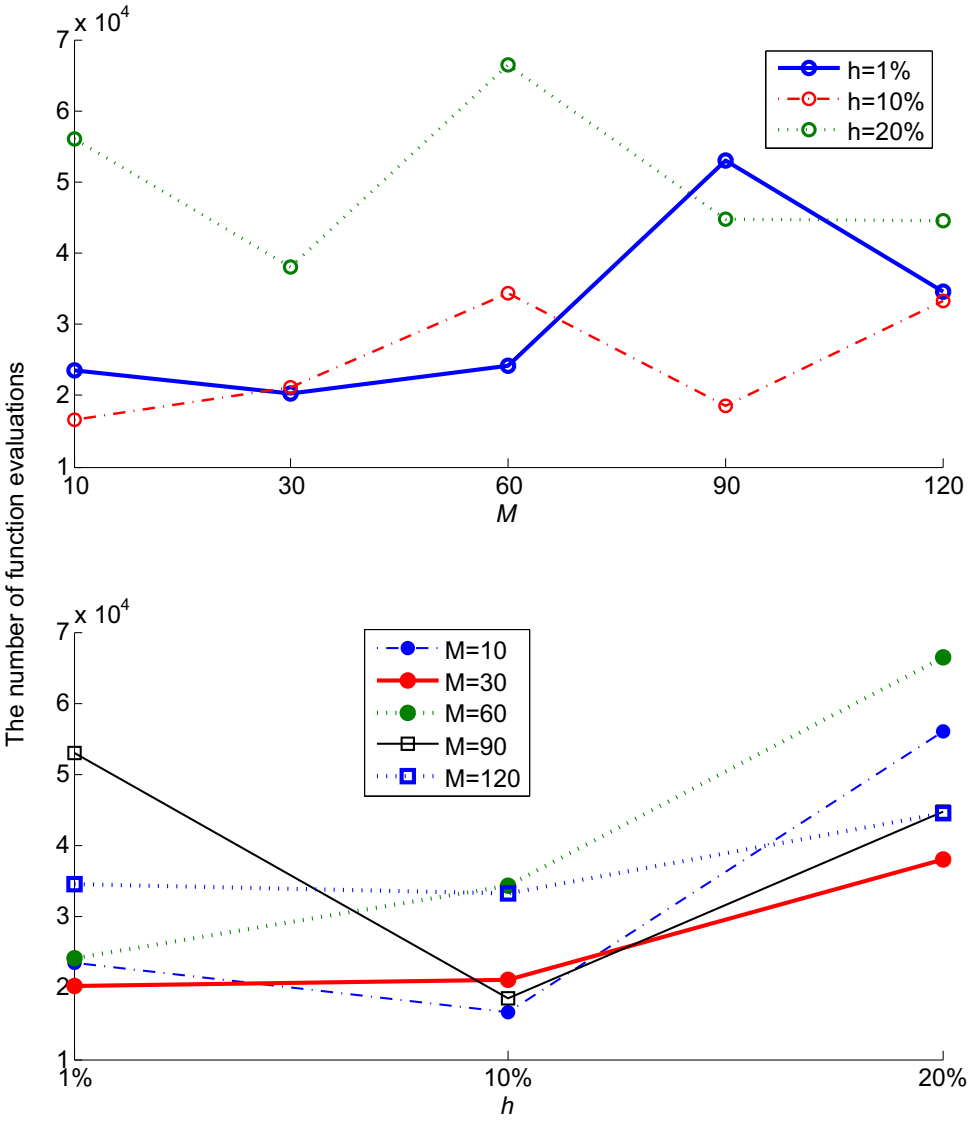


Fig. 30. Interaction effect plot for $M \times h$ for the sensor placement example (using the number of function evaluations)

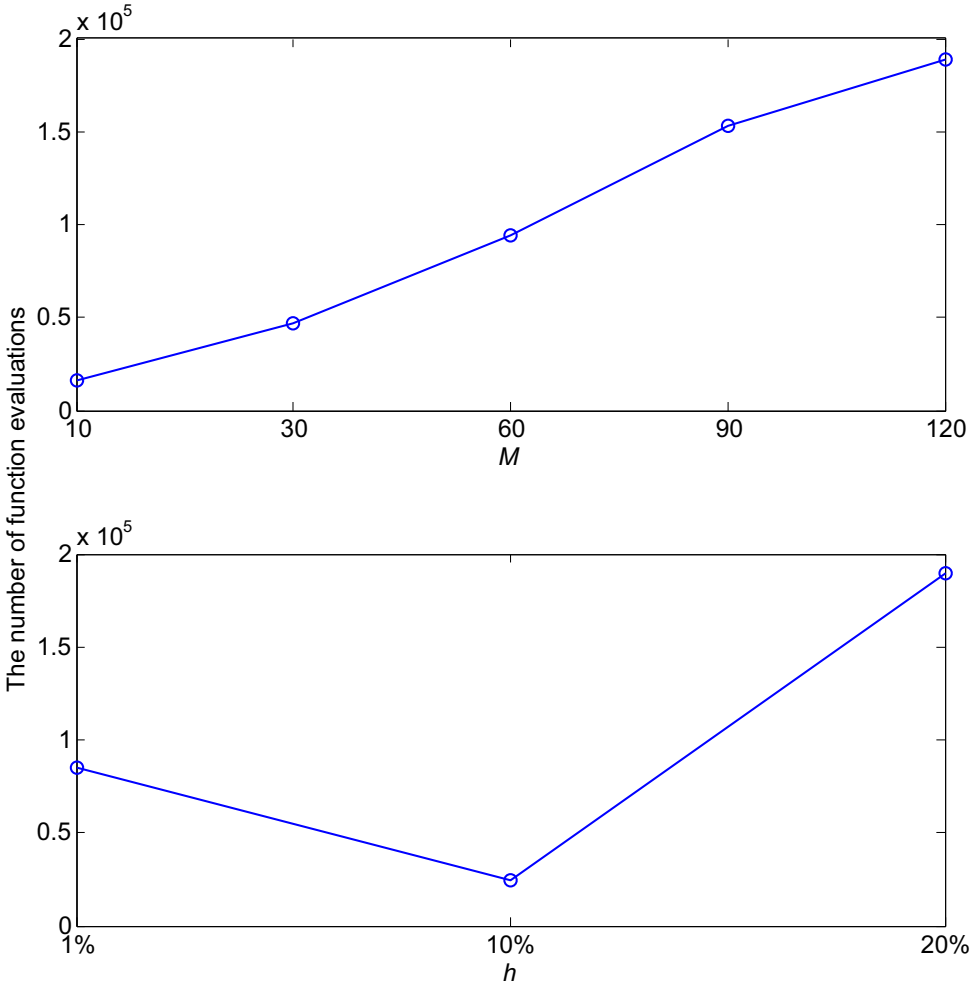


Fig. 31. Main effect plots for M and h for the Griewank example (using the number of function evaluations)

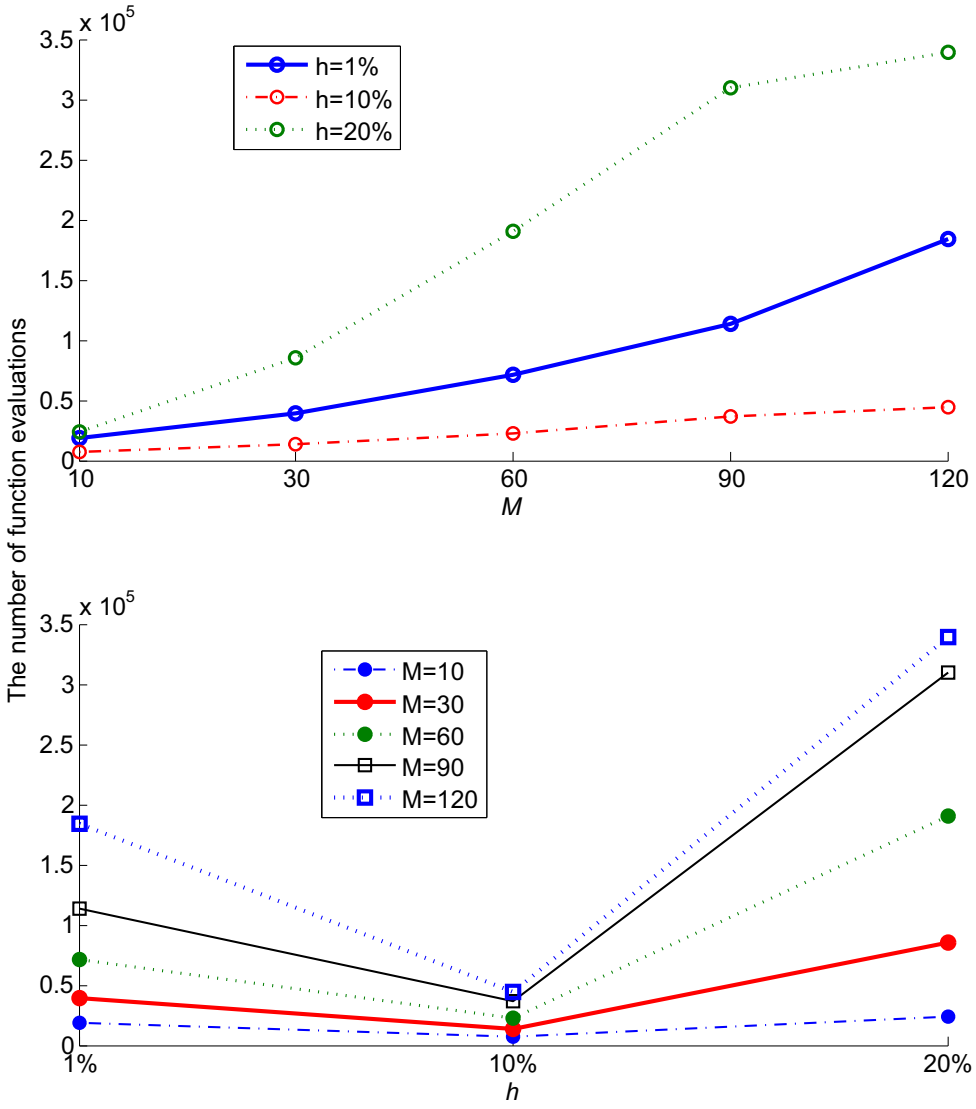


Fig. 32. Interaction effect plot for $M \times h$ for the Griewank example (using the number of function evaluations)

Based on our sensitivity analysis for both examples, we understand that the switching condition M and the percentile value h are the important factors affecting the performance of AEMC. Appropriate values of M and h are problem dependent, especially M. This is consistent with our intuition that M should be relatively larger for more complicated functions and relatively smaller for less complicated functions. We feel that for a more complicated function, EMC needs more time to gather representative samples for the use of data mining. We have observed that the value of h is especially critical to the performance of AEMC. When h is selected appropriately, the performance of AEMC is less dependent on the value of M. To choose suitable values for M and h, users may follow our general guidelines outlined in Section C.4 in Chapter III and further tune their values for specific problems.

We also studied the parameters whose values are relatively easier to set. These parameters are κ (the probability of sampling from the promising regions in the data-mining mode) and ρ (the parameter that controls the decreasing rate of P_k). The effects of these parameters are relatively straightforward to understand, as explained in Section C.4 in Chapter III. In the following, we set $M = 30$, $h = 10\%$, and $J_H = 6$.

Obviously, AEMC should mainly sample from the promising regions, i.e., κ should be close to 1. We have tested the performance of AEMC with $\kappa = .85$, .9, and .95. We ran AEMC for 10^5 function evaluations on both the Griewank example ($d = 50$) and the sensor placement example ($d = 9$), and we recorded the final objective function value. The results are shown in Table VI. As can be seen, the performances of AEMC with different κ values are very similar.

As to the value of ρ, it controls the decreasing rate of P_k. As we introduced in Chapter III.2, P_k is the probability of updating the rules learned by CART. Thus, ρ should be small enough so that P_k does not decrease too fast. Otherwise the probability of executing the "learning" in the data-mining mode decreases quickly to

Table VI. Effects of κ on the performance of AEMC

Example	$\kappa = .85$	$\kappa = .9$	$\kappa = .95$
Griewank	13.93	13.51	13.49
Sensor placement	1.24	1.25	1.25

Table VII. Effects of ρ on the performance of AEMC

Example	$\rho = .05$	$\rho = .1$	$\rho = .15$
Griewank	13.68	13.46	14.23
Sensor placement	1.25	1.25	1.24

0 and the AEMC algorithm virtually becomes EMC. We tested the performance of AEMC with $\rho = .05$, .1, and .15 on both the Griewank example ($d = 50$) and the sensor placement example ($d = 9$), and we also recorded the final objective function value. Table VII presents the results, which shows that the performance of AEMC is not sensitive to the value of ρ (given ρ is small). The variances in the algorithm performances are much smaller than those introduced by M and h (as shown in Fig. 27 and Fig. 28).

D. Sampling from a Mixture Gaussian Distribution

AEMC falls into the category of adaptive MCMC algorithms, and thus could be used to draw samples from a given target distribution. As we have proven in Chapter IV, the distribution of those samples will asymptotically converge to the target distribution.

When using AEMC for sampling purposes instead of optimization, some adjustments needed to be made. For optimization, AEMC should mainly focus on sub-regions that contain samples with better objective function values in order to

find better samples faster. For sampling, however, AEMC needs to gather samples all over the sample space so that the samples will indeed represent the target distribution. In other words, if the target distribution has multiple modes, AEMC should be able to gather samples from every mode when used for sampling, while it only needs to collect samples from a few "promising" (in terms of objective function value) modes when used for optimization.

Recall that AEMC uses the data-mining operations to adaptively learn information about the local optima of the objective function. For sampling, the data-mining operations are used to learn information about the modes of the target distribution. Thus, we need to adjust the h value in the data-mining operations. Specifically, h should be set to a relatively large value so that information about multiple modes could be learned. If h is too small, AEMC only focus on a few peaks of the target distribution. We recommend choosing $h = 25\%$.

We test AEMC on a five-dimensional mixture Gaussian distribution

$$\pi(\boldsymbol{x}) = \frac{1}{3}N_5(\mathbf{0}, I_5) + \frac{2}{3}N_5(\mathbf{5}, I_5),$$

where $\mathbf{0} = (0, 0, 0, 0, 0)$ and $\mathbf{5} = (5, 5, 5, 5, 5)$. This example is used in [27]. Since in this dissertation we assume the sample space to be bounded, we set the sample space to be $[-10, 10]^5$ here (it already covers almost all the probability mass). The distance between the two modes is $5\sqrt{5}$, which makes it difficult to jump from one mode to another. We compare the performance of AEMC with the Metropolis algorithm and EMC. Each algorithm was used to obtain 10^5 samples and all numerical results were averages of 10 runs.

The Metropolis algorithm was applied with a uniform proposal distribution $U[x-2, x+2]^5$. The acceptance rate was .22. The Metropolis algorithm could not escape from the mode in which it started. We then compare AEMC with EMC. We only

look at samples of the first dimension, since each dimension is independent of each other. The true histogram of the distribution is known so that we can calculate the L^2 distance between the estimated mass vector and the true distribution. Specifically, we divide the interval $[-10, 10]$ into 40 intervals (with a resolution of .5), and we can calculate the true and estimated probability mass respectively in each of the intervals.

All EMC related parameters are set following [27]. In AEMC, we set $h = 25\%$ so that samples from both modes can be obtained. If h is too small, AEMC will focus only on the peaks of the function and thus only samples around the mode $\mathbf{5}$ can be obtained (this is because the probability of sampling around the mode $\mathbf{5}$ is twice as large as the mode $\mathbf{0}$). In EMC, we employ the mutation and crossover operators used in [27]. The acceptance rates of mutation and crossover operators were .22 and .44, respectively. In AEMC, the acceptance rates of mutation, crossover, and data-mining operators were .23, .54, and .10, respectively. Fig. 33 shows the L^2 distance versus the number of samples for the three methods in comparison. AEMC converges faster than EMC and the Metropolis algorithm, and its sampling quality is far better than the Metropolis algorithm and it also achieves better sampling quality than EMC.

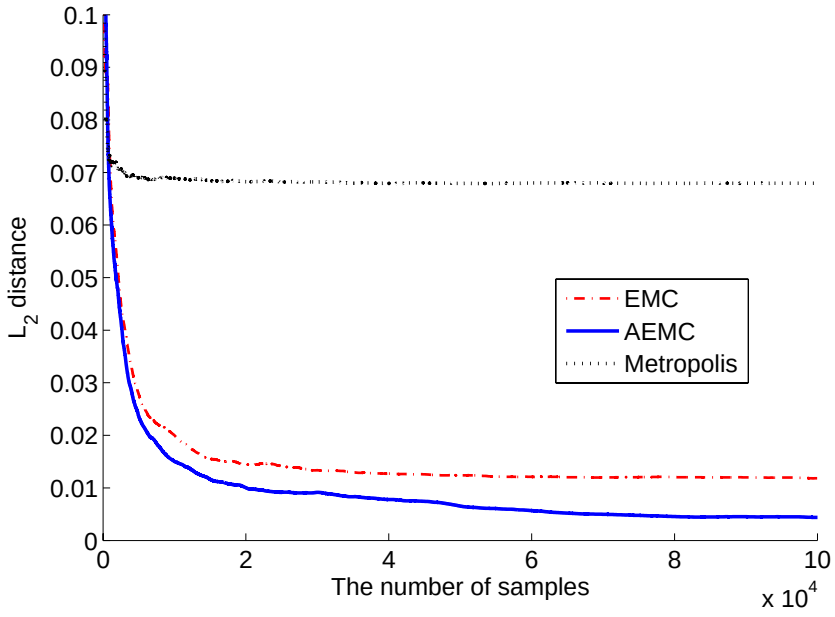


Fig. 33. Convergence rate of different algorithms

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

In this Chapter, this dissertation is concluded and its key findings are summarized. Furthermore, directions for future research are discussed.

A. Conclusions

This dissertation is motivated by engineering optimization, where a "black-box" objective function needs to be optimized. Without a clear structure of the objective function, it is difficult to find an algorithm that could take advantage of the problem structure and solve the optimization problem efficiently. This dissertation presents a stochastic search algorithm to solve such an optimization problem. The proposed algorithm combines random search techniques and metamodeling mechanisms. It learns promising search regions from previously visited solutions and sequentially improves the quality of the solutions. It does not require any information about the structure of the objective function, which makes it appropriate for many engineering optimization problems. The key findings of this dissertation are summarized as follows.

1. Combining Random Sampling and Metamodeling for Optimization

A general framework for combining *random sampling* methods and *metamodeling* methods is provided. The hybrid algorithm uses random sampling techniques to stochastically search over the sample space and gather representative samples that are representations of local optima. Based on previously visited samples, metamodeling methods are used to adaptively build a predictive model that points out where promising search regions are. Our numerical results show that such an algorithm could incorporate strengths from both methods and compensates for the limitations

of one another. In this dissertation, evolutionary Monte Carlo algorithm is used for random sampling purposes and CART is used for metamodeling purposes. Thus, a CART-guided adaptive evolutionary Monte Carlo algorithm is proposed. The AEMC algorithm is used to solve a sensor placement problem in a multi-station assembly process. Based on our numerical study, AEMC indeed improves the convergence speed compared to some other methods, including EMC, genetic algorithm, the stand-alone CART guided method, and the COMPASS algorithm. The AEMC algorithm is also tested on a suite of well-known test functions. AEMC has shown some potential to solve a general optimization problem.

2. Ergodicity of AEMC

The objective of the ergodicity study is to provide a theoretical guarantee that AEMC will eventually find the global optimum given enough run time. Since engineering optimization and many other optimization problems often involve objective functions with multiple local optima, having the capability of escaping local optima and eventually reaching the global optimum is critical for an effective search algorithm. Ergodicity of AEMC is studied and a theorem is stated and proved. Samples obtained by AEMC will be asymptotically distributed according to the Boltzmann distribution determined by the objective function. Therefore, AEMC will have larger probability of sampling from regions around the global optimum than sampling from other regions.

The proof of ergodicity does not depend on the specific algorithm used for random sampling and metamodeling. Therefore, users potentially could replace EMC and/or CART with other techniques. The algorithm proposed in this dissertation could be treated as a general framework for constructing effective optimization algorithms.

3. Parameter Selections for AEMC and Their Sensitivity

An numerical study is presented to provide some understandings about the effects of tuning parameters in AEMC. Two parameters, the switching condition M and the percentile value h, are found to be critical to the performance of AEMC. An appropriate value of M is problem dependent. The value of M could be determined by the number of new samples obtained between two data-mining (or metamodeling) modes, i.e., M could be chosen based on nM where n is the population size. From our experience, $nM \approx 150 \sim 450$ works well. The value of h should be moderately small (i.e., $5\% \sim 15\%$). If h is too large, the learned "promising" regions are too broad and thus AEMC converges slowly. On the other hand, if h is too small, AEMC focuses too much on the currently known peaks of the objective function surface and may miss some interesting regions.

4. AEMC for Sampling

AEMC also falls into the category of adaptive Markov chain Monte Carlo (MCMC) algorithms and is the first adaptive MCMC algorithm that simulates multiple Markov chains in parallel. Therefore, it could use information from multiple Markov chains to improve algorithm performance. The EMC algorithm itself has already been shown to be an enhanced tool as a sampling method, and the data-mining component in AEMC further improves the convergence rate to the target distribution without destroying the ergodicity of the algorithm. We validated the effectiveness of AEMC for sampling purposes using a mixture Gaussian distribution. From the numerical results, it is shown that data-mining operations could adaptively construct better proposal distributions so that samples obtained from AEMC converge faster to the target distribution than non-adaptive MCMC methods, such as Metropolis algorithm

and evolutionary Monte Carlo algorithm.

B. Future Work

While it provides useful findings and insights, this dissertation also provides a foundation for future study.

1. Applications to Other Engineering Optimization and Sampling

For the sensor placement problem in this dissertation, it is assumed that the number of sensors is pre-determined. When the sensor number is not known, a two-step procedure is warranted to obtain a design of sensor placement. Firstly, one needs to specify a range for the sensor number and randomly select a number from that range as the sensor number. Then, given the sensor number, one could obtain a design of sensor placement as shown in this dissertation. The effectiveness of such a strategy needs to be studied.

The effectiveness of AEMC in ultra-high dimensional problems still need to be investigated. From the numerical results shown in this dissertation, it is observed that the performance of AEMC does not deteriorate as the problem dimension increases from 2 to 50. Yet it is worthwhile to study the performance of AEMC for even higher dimensional problems.

This dissertation has conducted a preliminary study on AEMC's effectiveness for sampling purposes. More numerical analysis should be done to investigate the usage of AEMC for sampling. Sampling from higher dimensional distributions (e.g., distributions used in [26] and [27]) should be tested. Furthermore, AEMC should be tested in some real world applications such as Bayesian mixture models [27] and protein folding simulations [71].

2. Search Space Annealing

As shown in Theorem 2, AEMC is able to find the global optimum given long enough run time. However, due to the high dimensionality and broadness of the sample space, the process may still be slow. To accelerate the search process, we could shrink the search space over iterations toward regions that contain better solutions. CART partitions the sample space into different regions and identifies "promising" and "unpromising" regions. Using this information, we could shrink the search space toward the "promising" regions and not search in the "unpromising" regions. However, because of the model fitting error in the CART method, the so-called "promising" and "unpromising" regions might be inaccurate or even misleading. Therefore, a search algorithm needs to keep a history of the space shrinkage and be able to go back to previous larger search spaces. Furthermore, theoretical study should be conducted to study if the space annealing still guarantees that the search algorithm reaches the global optimum when given enough time.

3. Metamodeling Procedures

Since the ergodicity of AEMC does not depend on the specific method used for metamodeling, users have the freedom to choose a powerful and suitable data mining method in the data mining mode of AEMC. In the current version of AEMC, we use CART as the data-mining method and the sample space is sliced into hyper-rectangles. When the function surface is complex, a rectangular partition may not be sufficient. A more sophisticated partition may be required. However, a good replacement for CART may not be straightforward to find because any viable candidate must be computationally efficient so as not to slow down the optimization process. Some other data-mining methods, such as artificial neural networks, may have more

"learning" power, but they are much more computationally expensive themselves and are therefore less likely to be a good candidate for the AEMC algorithm. Some potential choices are other tree-based methods such as QUEST (quick, unbiased and efficient statistical tree) [72] and CRUISE (classification rule with unbiased interaction selection and estimation) [73]. In their numerical study, Loh and Shih [72] showed that QUEST could be faster and more accurate than CART. CRUISE allows for multivariate partitions and thus could construct more flexible regions.

4. Other Extensions

Incorporating structural information of an objective function. In this dissertation, we assume that the objective function is a "black-box" function and no structural information is known. Yet some structural information could be available based on engineering knowledge or other expert opinions. Taking the sensor placement problem for an example, engineering knowledge may suggest that two sensors not be put too close to each other, otherwise sensor measurements might be redundant. It is worthwhile to study how to incorporate available (while probably limited) information of the objective function into an optimization algorithm to improve algorithm performance.

Handling computationally expensive objective functions. In many real-world applications, objective functions could be very computationally expensive, e.g., evaluating one solution could take a few hours when a finite element analysis (FEA) is required. Therefore, one can only afford to evaluate a very limited number of solutions. Under these circumstances, a more greedy algorithm is needed. One will focus on the algorithm performance at an early stage, i.e., the optimization algorithm should improve the quality of a solution very fast even

though the quality of the final result is sacrificed.

Extensions to stochastic simulation-based optimization. An interesting extension is to apply the idea of AEMC to simulation-based optimization, which considers the problem of optimizing stochastic objective functions. Due to the stochastic nature, the objective function values need to be estimated instead of evaluated deterministically. Therefore, additional considerations are needed to modify the deterministic version of AEMC, i.e., the algorithm should maintain appropriate balance among searching the entire sample space, searching local "promising" regions, and improving estimates of objective function values.

REFERENCES

[1] W. Roush, A. M. Goho, E. Scigliano, D. Talbot, M. M. Waldrop, G. T. Huang, P. Fairley, E. Jonietz, and H. Brody, "10 emerging technologies that will change the world," *Technol. Rev.*, vol. 106, pp. 33–49, Feb 2003.

[2] S. S. Mandroli, A. K. Shrivastava, and Y. Ding, "A survey of inspection strategy and sensor distribution studies in discrete-part manufacturing processes," *IIE Trans.*, vol. 38, pp. 309–328, 2006.

[3] S. T. S. Bukkapatnam, J. M. Nichols, M. Seaver, S. T. Trickey, and M. Hunter, "A wavelet-based, distortion energy approach to structural health monitoring," *Struct. Health Monitor. J.*, vol. 4, pp. 247–258, 2005.

[4] A. Čivilis, C. S. Jensen, and S. Pakalnis, "Techniques for efficient tracking of road-network-based moving objects," *IEEE Trans. Knowl. Data Eng.*, vol. 17, pp. 698–712, 2005.

[5] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed target classification and tracking in sensor networks," *Proc. IEEE*, vol. 91, pp. 1163–1171, 2003.

[6] P. Kim and Y. Ding, "Optimal engineering system design guided by data-mining methods," *Technometrics*, vol. 47, pp. 336–348, 2005.

[7] S. Ólafsson and J. Kim, "Simulation optimization," in *Proc. 2002 Winter Simulation Conf.*, 2002, San Diego, CA, Dec., pp. 79–84.

[8] M. C. Fu, "Optimization for simulation: theory vs. practice," *INFORMS J. Comput.*, vol. 14, pp. 192–215, 2002.

[9] J. April, F. Glover, J. P. Kelly, and M. Laguna, "Practical introduction to simulation optimization," in *Proc. 2003 Winter Simulation Conf.*, 2003, New Orleans, LA, Dec., pp. 71–78.

[10] F. Azadivar, "Simulation optimization methodologies," in *Proc. 1999 Winter Simulation Conf.*, 1999, Pheoenix, AZ, Dec., pp. 93–100.

[11] J. R. Swisher, P. D. Hyden, S. H. Jacobson, and L. W. Schruben, "A survey of simulation optimization techniques and procedures," in *Proc. 2000 Winter Simulation Conf.*, 2000, Orlando, FL, Dec., pp. 119–128.

[12] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete Event Systems Simulation*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 2000.

[13] L. J. Hong and B. L. Nelson, "Discrete optimization via simulation using COMPASS," *Oper. Res.*, vol. 54, pp. 115–129, 2006.

[14] K. G. Murty, *Linear Programming*, New York: John Wiley & Sons, 1983.

[15] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*, 2nd ed. New York: John Wiley & Sons, 2001.

[16] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Stat. Sci.*, vol. 8, pp. 10–15, 1993.

[17] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, Cambridge, MA: MIT Press, 1992.

[18] W.H. Wong and F. Liang, "Dynamic Weighting in Monte Carlo and Optimization," *P. Natl. Acad. Sci. USA*, vol. 94, pp. 14220–14224, 1997.

[19] V. C. P. Chen, K. Tsui, R. R. Barton, and M. Meckesheimer, "A review on design, modeling and applications of computer experiments," *IIE Trans.*, vol. 38, pp. 273 – 291, 2006.

[20] K. Fang, R. Li, and A. Sudjianto, *Design and Modeling For Computer Experiments*, London, UK: Chapman & Hall/CRC, 2006.

[21] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Stat. Sci.*, vol. 4, pp. 409–435, 1989.

[22] T. W. Simpson, J. Peplinski, P. N. Koch, and J. K. Allen, "On the use of statistics in design and the implications for deterministic computer experiments," in *Proc. DETC'97, 1997 ASME Design Engineering Tech. Conf.*, 1997, Sacramento, CA, Sept., Paper No. DETC97/DTM-3881.

[23] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*, New York: Springer-Verlag, 2003.

[24] H. Liu and T. Igusa, "Feature-based classification for design optimization," *Res. Eng. Des.*, vol. 17, pp. 189–206, 2007.

[25] M. Schwabacher, T. Ellman, and H. Hirsh, "Learning to set up numerical optimizations of engineering designs," *AI EDAM*, vol. 12, pp. 173–192, 1998.

[26] F. Liang and W. H. Wong, "Evolutionary Monte Carlo: applications to C_p model sampling and change point problem," *Stat. Sinica*, vol. 10, pp. 317–342, 2000.

[27] F. Liang and W. H. Wong, "Real-parameter evolutionary Monte Carlo with applications to bayesian mixture models," *J. Am. Stat. Assoc.*, vol. 96, pp. 653–666, 2001.

[28] S. Geman and D. Geman, "Stochastic relaxation, gibbs distribution and the bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 6, pp. 721–741, 1984.

[29] F. Liang, "A generalized Wang-Landau algorithm for Monte Carlo computation," *J. Am. Stat. Assoc.*, vol. 100, pp. 1311–1327, 2005.

[30] F. Liang, C. Liu, and R. J. Carroll, "Stochastic approximation in Monte Carlo computation," *J. Am. Stat. Assoc.*, vol. 102, pp. 305–320, 2007.

[31] R. M. Neal, "Bayesian learning for neural networks," in *Lecture Notes in Statistics*, No. 118. New York: Springer-Verlag, 1996.

[32] W. R. Gilks G. O. Roberts and S. K. Sahu, "Adaptive Markov chain Monte Carlo through regeneration," *J. Am. Stat. Assoc.*, vol. 93, pp. 1045–1054, 1998.

[33] A. E. Brockwell and J. B. Kadane, "Identification of regeneration times in MCMC simulation, with application to adaptive schemes," *J. Comput. Graph. Stat.*, vol. 14, pp. 436–458, 2005.

[34] H. Haario, E. Saksman, and J. Tamminen, "An adaptive metropolis algorithm," *Bernoulli*, vol. 7, pp. 223–242, 2001.

[35] C. Andrieu and C.P. Robert, "Controlled MCMC for optimal sampling," [Online]. Available: http://www.ceremade.dauphine.fr/∼xian/control.ps.gz, 2001.

[36] Y. F. Atchadé and J. S. Rosenthal, "On adaptive Markov chain Monte Carlo algorithms," *Bernoulli*, vol. 11, pp. 815–828, 2005.

[37] G. O. Roberts and J. S. Rosenthal, "Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms," *J. Appl. Probab.*, vol. 44, pp. 458–475, 2007.

[38] G. E. P. Box and K. B. Wilson, "On the experimental attainment of optimum conditions," *J. Roy. Stat. Soc. B Met.*, vol. 13, pp. 1–45, 1951.

[39] J. P. C. Kleijnen, *Statistical tools for Simulation Practitioners*, New York: Marcel Dekker, 1987.

[40] S. D. Guikema, R. A. Davidson, and Z. Çağnan, "Efficient simulation-based discrete optimization," in *Proc. 2004 Winter Simulation Conf.*, 2004, Washington, D.C., Dec., pp. 536–544.

[41] A. L. Huyet, "Optimization and analysis aid via data-mining for simulated production systems," *Eur. J. Oper. Res.*, vol. 173, pp. 827–838, 2006.

[42] R. S. Michalski, "Learnable evolution model: evolutionary processes guided by machine learning," *Mach. Learn.*, vol. 38, pp. 9–40, 2000.

[43] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, New York: McGraw Hill, 2001.

[44] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, pp. 308C313, 1965.

[45] K.-T. Fang and Y. Wang, *Number-Theoretic Methods in Statistics*, New York: Chapman & Hall, 1994.

[46] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, "Introducing Markov chain Monte Carlo," in *Markov Chain Monte Carlo in Practice*, W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, Eds. London, UK: Chapman & Hall, 1995, pp. 1–19.

[47] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, New York: Chapman & Hall, 1984.

[48] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equations of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, pp. 1087–1091, 1953.

[49] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, pp. 97–109, 1970.

[50] L. Tierney, "Markov chains for exploring posterior distributions," *Ann. Stat.*, vol. 22, pp. 1701–1762, 1994.

[51] S.P. Meyn and R.L. Tweedie, *Markov Chains and Stochastic Stability*, London, UK: Springer-Verlag, 1993, [Online]. Available: http://probability.ca/MT/.

[52] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, New York: Springer-Verlag, 2001.

[53] P. Müller, "A generic approach to posterior integration and Gibbs sampling," Department of Statistics, Purdue University, West Lafayette, IN, Technical Report 91-09, 1991.

[54] K. L. Mengersen and R. L. Tweedie, "Rates of convergence of the Hastings and Metropolis algorithms," *Ann. Stat.*, vol. 24, pp. 101–121, 1996.

[55] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.

[56] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Ann. Stat.*, vol. 29, pp. 1189–1232, 2001.

[57] H. Bersini, M. Dorigo, S. Langerman, G Geront, and L. Gambardella, "Results of the first international contest on evolutionary optimisation," in *Proc. IEEE Int. Conf. Evolutionary Computation*, 1996, Nagoya, Japan, May, pp. 611–615.

[58] H. Mühlenbein, D. Schomisch, and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel Computing*, vol. 17, pp. 619–632, 1991.

[59] A. O. Griewank, "Generalized descent for global optimization," *J. Optimiz. Theory App.*, vol. 34, pp. 11–39, 1981.

[60] J. Jin and J. Shi, "State space modeling of sheet metal assembly for dimensional control," *J. Manuf. Sci. E.-T. ASME*, vol. 121, pp. 756–762, 1999.

[61] Y. Ding, D. Ceglarek, and J. Shi, "Modeling and diagnosis of multistage manufacturing processes: Part I - state space model," in *Proc. 2000 Japan/USA Symp. Flexible Automation*, Ann Arbor, MI, 2000.

[62] J. Camelio, S. J. Hu, and D. Ceglarek, "Modeling variation propagation of multi-station assembly systems with compliant parts," *J. Mech. Design*, vol. 125, pp. 673–681, 2003.

[63] D. Djurdjanovic and J. Ni, "Dimensional errors of fixtures, locating and measurement datum features in the stream of variation modeling in machining," *J. Manuf. Sci. Eng.*, vol. 125, pp. 716–730, 2003.

[64] Q. Huang, J. Shi, and J. Yuan, "Part dimensional error and its propagation modeling in multi-operational machining processes," *J. Manuf. Sci. E.-T. ASME*, vol. 125, pp. 255–262, 2003.

[65] S. Zhou, Q. Huang, and J. Shi, "State space modeling of dimensional variation propagation in multistage machining process using differential motion vectors," *IEEE Trans. Robot. Autom.*, vol. 19, pp. 296–309, 2003.

[66] C. Liu, Y. Ding, and Y. Chen, "Optimal coordinate sensor placements for estimating mean and variance components of variation sources," *IIE Trans.*, vol.

37, pp. 877–889, 2005.

[67] J. R. Schott, *Matrix Analysis for Statistics*, New York: John Wiley & Sons, 1997.

[68] F. Pukelsheim, *Optimal Design of Experiments*, New York: John Wiley & Sons, 1993.

[69] Y. Ding, J. Shi, and D. Ceglarek, "Diagnosability analysis of multi-station manufacturing processes," *J. Dyn. Syst.-T. ASME*, vol. 124, pp. 1–13, 2002.

[70] C. F. J. Wu and M. Hamad, *Experiments: Planning, Analysis, and Parameter Design Optimization*, New York: John Wiley & Sons, 2000.

[71] F. Liang and W. H. Wong, "Evolutionary Monte Carlo for protein folding simulations," *J. Chem. Phys.*, vol. 115, pp. 3374–3380, 2001.

[72] W.-Y. Loh and Y.-S. Shih, "Split selection methods for classification trees," *Statistica Sinica*, vol. 7, pp. 815–840, 1997.

[73] H. Kim and W.-Y. Loh, "Classification trees with unbiased multiway splits," *J. Amer. Statist. Assoc.*, vol. 96, pp. 598–604, 2001.

APPENDIX A

VEC OPERATOR AND HADAMARD PRODUCT

The definitions of vec operator and Hadamard product are given in [[67]]. Vec operator is used to transform a matrix to a vector that has elements of the matrix as its elements. If a matrix $\boldsymbol{V}^{q\times w}$ has $\boldsymbol{v}_i$ as its i-th column vector, then the vec($\boldsymbol{V}$) is a $qw \times 1$ vector given by

$$\text{vec}(\boldsymbol{V}) = \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_w \end{pmatrix}.$$

A property of vec operator is given by Theorem 8.10 in [67]. Let $\boldsymbol{U}$ and $\boldsymbol{V}$ both be $q \times w$ matrices. Then

$$\text{tr}(\boldsymbol{U}^T\boldsymbol{V}) = \{\text{vec}(\boldsymbol{U})\}^T\text{vec}(\boldsymbol{V}). \tag{A.1}$$

The Hadamard product, denoted by $\otimes$, simply performs the element-wise multiplication of two matrices. Given two matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ that are each $q \times w$, then

$$\boldsymbol{U} \otimes \boldsymbol{V} = \begin{pmatrix} u_{11} & \cdots & u_{1w} \\ \vdots & \ddots & \vdots \\ u_{q1} & \cdots & u_{qw} \end{pmatrix} \otimes \begin{pmatrix} v_{11} & \cdots & v_{1w} \\ \vdots & \ddots & \vdots \\ v_{q1} & \cdots & v_{qw} \end{pmatrix} = \begin{pmatrix} u_{11}v_{11} & \cdots & u_{1w}v_{1w} \\ \vdots & \ddots & \vdots \\ u_{q1}v_{q1} & \cdots & u_{qw}v_{qw} \end{pmatrix}.$$

APPENDIX B

DERIVATION OF SENSITIVITY FUNCTION

From the property of vec operator given in (A.1), we have

$$\begin{aligned}\operatorname{tr}\left((\delta\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}})^T\delta\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}}\right) &= \left(\operatorname{vec}(\delta\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}})\right)^T \operatorname{vec}(\delta\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}}) \\ &= \left(\boldsymbol{\pi}(\boldsymbol{\Psi})\cdot\delta\boldsymbol{\sigma}\right)^T\left(\boldsymbol{\pi}(\boldsymbol{\Psi})\cdot\delta\boldsymbol{\sigma}\right) \qquad \text{(according to (5.5))} \\ &= (\delta\boldsymbol{\sigma})^T\left(\boldsymbol{\pi}(\boldsymbol{\Psi})^T\boldsymbol{\pi}(\boldsymbol{\Psi})\right)\delta\boldsymbol{\sigma}\end{aligned}$$

Therefore the sensitivity function is

$$\begin{aligned}S &= \min_{\delta\boldsymbol{\sigma}\neq 0}\frac{\operatorname{tr}\left((\delta\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}})^T\delta\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}}\right)}{(\delta\boldsymbol{\sigma})^T(\delta\boldsymbol{\sigma})} \\ &= \min_{\delta\boldsymbol{\sigma}\neq 0}\frac{(\delta\boldsymbol{\sigma})^T\left(\boldsymbol{\pi}(\boldsymbol{\Psi})^T\boldsymbol{\pi}(\boldsymbol{\Psi})\right)\delta\boldsymbol{\sigma}}{(\delta\boldsymbol{\sigma})^T(\delta\boldsymbol{\sigma})}.\end{aligned}$$

Using the eigenvalue property of a symmetric matrix (refer to Theorem 3.16 in [67]), it is straightforward to see that

$$S = \lambda_{\min}\left(\boldsymbol{\pi}(\boldsymbol{\Psi})^T\boldsymbol{\pi}(\boldsymbol{\Psi})\right).$$

VITA

Mr. Yuan Ren received his B.E. degree in Automation from Tsinghua University, Beijing, China in 2003. In September 2003, he enrolled in the doctoral program in the Department of Industrial and Systems Engineering at Texas A&M University, College Station, TX. His research interests are in the area of quality engineering, applied statistics and applied optimization, including analysis and design of distributed sensor systems for quality improvement, data mining methods, and industrial experimental designs. He is a student member of IIE and INFORMS. His permanent address is No.121 Jinyang Road, Chengdu, Sichuan 610000, China.